z/XDC®
INSTALLATION GUIDE

z/XDC® Release z1.13 for z/OS

David B. Cole

PREFACE

PROPRIETARY LEGEND

z/XDC® and its documentation (collectively, "Product"), including copies thereof, are the confidential and proprietary property of ColeSoft Partners, Inc ("Owner"). Product may be used only by those organizations that are licensed by Owner for such use and only in the manner so licensed. The program and documentation may not be published, reproduced, distributed, or made available to third parties for any purpose without the expressed written permission of Owner; however, a reasonable number of copies may be made of the documentation (including the copyright notices and proprietary legends thereon) as is necessary for the legitimate use of Product within a licensed organization.

Except as may be otherwise expressed in a signed agreement between Owner and Customer, Owner makes 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 programs and subroutines. 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 be liable for any loss or damage whatsoever (including death) arising from Product, whether such loss or damage be direct, indirect, consequential, special or otherwise. Further, Owner shall not be obligated to indemnify any user of Product in any manner for any loss which the user or anyone else may experience, of any kind or nature, arising out of the use or misuse of Product.

CONTACTING COLESOFT

The **z/XDC®** products are marketed by **ColeSoft Marketing, Inc** with its principal office in Afton, Virginia. If you want more information, please contact ColeSoft as follows:

Phone:	**928-771-2003**
Toll Free:	**800-XDC-5150**
FAX:	**928-771-2005**
E-Mail:	**sales@colesoft.com**
Home Page:	**http://www.colesoft.com**

Our Technical Support contacts are:

Phone:	**540-456-8210**
FAX:	**540-456-6658**
E-Mail:	**techsupt@colesoft.com**
Home Page:	**http://www.colesoft.com**

FTP site: **ftp.colesoft.com**

Our Customer Services contacts are:

Phone: **540-456-8210**
FAX: **540-456-6658**
E-Mail: **support@colesoft.com**
Home Page: **http://www.colesoft.com**

Our snail mail address is:

Address: **ColeSoft**
736 Fox Hollow Road
Afton, Virginia 22920
USA

Our home page 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.
- FTP links for downloading current maintenance for z/XDC.
- Links permitting existing customers to download a full set of z/XDC's documentation.
- Online product delivery.
- 24x7 self-service for temporary, short-term, license activation codes for use in D.R. tests and other emergencies.

TRADEMARKS

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) printable copies of all of z/XDC's manuals. Each manual is available in PDF format.

z/XDC® z1.13 INSTALLATION GUIDE

(Preface)

CONTENTS

FIGURES

z/XDC® z1.13 INSTALLATION GUIDE

THE z/XDC PRODUCT DISTRIBUTION PACKAGE

z/XDC "Product Distribution Packages" can be obtained via Downloads from our web site: www.colesoft.com. A package consists of a compressed file that must be downloaded to your PC. There, you must unZIP the package. Portions of the unZIP'd result must remain on your PC, while other portions must then be uploaded to your mainframe to be decompressed (again, but this time via AMATERSE) and then installed (via SMP/E).

The Product Distribution Package is a ZIP'd file named *xdcz1d.zip*. It contains the following file structure (as shown in **Figure 1** on page 1). You will have to extract the files using PKUNZIP[1], WinZip[2], ZTree[3] or whatever decompression tool you prefer.

- The *software* folder contains an *xdcxmit.trs* file that will have to be uploaded to your mainframe and then restored to its original format via MVS's AMATERSE program (a.k.a. TRSMAIN) and then via TSO's RECEIVE[4] commands.

- The *manuals* folder contains all of z/XDC's manuals in PDF format.[5] The documentation can be searched, read, or printed directly from your hard drive using Adobe's Acrobat Reader.

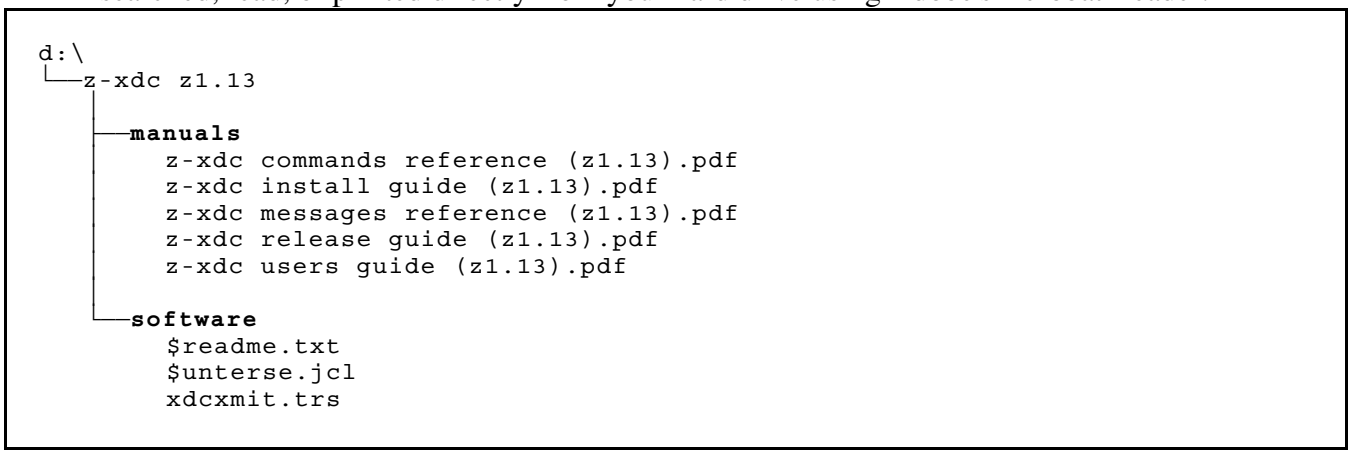

Figure 1 z/XDC Distribution Package Map

[1] PKZIP and PKUNZIP are file compression and decompression programs that are available from **www.pkware.com**.

[2] WinZip is a file compression/decompression utility that is available from **www.winzip.com**.

[3] ZTree is a superb file manager utility that is available from **www.ztree.com**. It includes a comprehensive interface to the compression/decompression utility of your choice.

[4] Do not confuse TSO's RECEIVE command with SMP/E's RECEIVE command. They are entirely different from each other.

[5] PDF stands for "Portable Document Format". It is a documentation format that was developed by and is supported by Adobe Systems, Inc. A reader for this format is available without charge from Adobe's web site: **www.adobe.com**. The name of the reader is "The Adobe Acrobat Reader".

z/XDC® z1.13 INSTALLATION GUIDE

(The z/XDC Product Distribution Package)

Printing z/XDC's Various Manuals

z/XDC's documentation consists of the manuals shown in **Figure 2** on page 2. The manuals are provided in **PDF** format. This is a good format for both browsing, searching, and printing.

z/XDC z1.13 Manual Name	Description
`Release Guide`	Describes what is new with release z1.13 of z/XDC.
`User's Guide`	Explains how to use z/XDC z1.13.
`Commands Reference`	Describes all of z/XDC z1.13's commands.
`Messages Reference`	Explains all numbered messages issued by z/XDC z1.13.
`Install Guide`	Describes how to install z/XDC z1.13 (the current document).

Figure 2 z/XDC Manuals

Printing the PDF-Formatted Manuals

PDF stands for "Portable Document Format". It is a documentation format that was developed by and is supported by Adobe Systems, Inc. A reader for this format is available without charge from Adobe's web site: www.adobe.com. The name of the reader is "The Adobe Acrobat Reader".

To print a z/XDC manual, start up Acrobat Reader and use it to open the desired manual, and then use the Reader's PRINT dialogs to print the manual. If possible, please be sure to select duplex printing.

Note, when you open a manual, Acrobat may issue the following warning message:

> *Unable to find or create the font 'WPTypographicSymbols'. Some characters may not display or print correctly.* ***OK***

This message occurs because the original z/XDC documents were created using WordPerfect, and Adobe apparently is not licensed to display some of the fonts that WordPerfect automatically uses. Simply press the **OK** button and proceed.

You may be able to correct this "problem" by selecting Acrobat Reader's **View** menu and then making sure that **Use Local Fonts** is checked. If you have WordPerfect installed on your system, this should take care of it.

If checking **Use Local Fonts** does not resolve the problem, then don't bother worrying about it any further. The "display problems" referred to by the warning are not really significant in any case.

Getting the Product Files to Your Mainframe

The distribution package's *software* folder contains three files: *$readme.txt*, *$unterse.jcl* and *xdcxmit.trs*.

- *xdcxmit.trs* contains the product's installation libraries in tersed xmit-library format.
- *$unterse.jcl* contains sample JCL for decompressing the xmit-library.
- *$readme.txt* identifies the release level of the *xdcxmit.trs* file.

You will now have to upload *xdcxmit.trs* to your mainframe and extract the installation libraries from with it. Here's how to do that.

Step 1: Transmit the *Xdcxmit.trs* Files to Your Mainframe

I assume that you are using a PC that is running some sort of "workstation" program that allows you to connect to your mainframe as a 327x type terminal. **Figure 3** on page 3 shows just a few examples of many such programs. These programs all have a file transfer capability for sending files back and forth between the mainframe and your PC.

Company	Workstation Product	Web Site
Attachmate	Extra!	www.attachmate.com
Ericom	PowerTerm	www.ericom.com
HOB	HOBLink Terminal Edition	www.hobsoft.com/produkte/classemu.jsp
OpenText	Host Explorer	connectivity.opentext.com/products/terminal-emulation.aspx
IBM	Personal Communications	www-01.ibm.com/software/network/pcomm
Micro Focus	Rumba	www.microfocus.com/products/RUMBA
Tom Brennan Software	Vista TN3270	www.tombrennansoftware.com

Figure 3 Commercial Workstation Software

At this point you will need to use either your workstation program's file transfer feature or an FTP dialog (run either from your PC or at the mainframe) to "upload" (i.e. send) the *xdcxmit.trs* file from your PC to your mainframe. When you perform the upload, please follow these guidelines:

- You must perform a "binary format" file transfer. This means that the transfer process **must not** attempt to translate the file from ASCII to EBCDIC, and it **must not** attempt to interpret CRLF ("carriage return, line feed") sequences as end-of-record signals.

- The *xdcxmit.trs* file must be uploaded to a mainframe dataset having the following characteristics:
  Suggested dsname: `DBCOLE.XDCZ1D.INSTALL.XDCXMIT.TERSED`
  DCB: `DSORG=PS,RECFM=FB,LRECL=1024,BLKSIZE=0`
  SPACE: `(TRK,(500,100),RLSE)`

Step 2: Decompress the XDCXMIT Library

`DBCOLE.XDCZ1D.INSTALL.XDCXMIT.TERSED` is a compressed copy of z/XDC's "pre-installation" library named `DBCOLE.XDCZ1D.INSTALL.XDCXMIT`. To decompress this library, you will need to run the JCL shown in **Figure 4** on page 4. A copy of this JCL is available in the distribution package's *product* folder, in a file named *$unterse.jcl*.

```
//$UNTERSE JOB (CSW,UPS),'Phone: 540-456-8536',CLASS=A,MSGCLASS=A,
//             MSGLEVEL=(1,1),NOTIFY=R9999,TIME=1439
//*
//*****************************************************************
//*                                                               *
//* This JCL can be used to decompress z/XDC's XDCXMIT library.   *
//*                                                               *
//*****************************************************************
//* First, insure that the output library is purged.              *
//*****************************************************************
//*
//RESET   EXEC PGM=IEFBR14
//XDCXMIT  DD  DSN=DBCOLE.XDCZ1D.INSTALL.XDCXMIT,
//             UNIT=DISK,SPACE=(TRK,0),DISP=(MOD,DELETE)
//*
//*
//*
//*****************************************************************
//* Now, build the z/XDC pre-installation library.                *
//*****************************************************************
//*
//UNTERSE EXEC PGM=AMATERSE,PARM=UNPACK,REGION=0M
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=DBCOLE.XDCZ1D.INSTALL.XDCXMIT.TERSED,DISP=SHR
//SYSUT2   DD  DSN=DBCOLE.XDCZ1D.INSTALL.XDCXMIT,
//             UNIT=SYSALLDA,DISP=(,CATLG),
//             SPACE=(TRK,(1400,500,5),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
```

Figure 4 Sample JCL for Decompressing `DBCOLE.XDCZ1D.INSTALL.XDCXMIT`

Step 3: Extract the Installation Libraries from `DBCOLE.XDCZ1D.INSTALL.XDCXMIT`

`DBCOLE.XDCZ1D.INSTALL.XDCXMIT` is a partitioned dataset most of whose members are xmit files (i.e. files created by TSO's XMIT command). These members need to be processed by the TSO's RECEIVE[4] command to produce the z/XDC installation libraries. Information about these libraries is shown in **Figure 5** on page 5.

The Pre-Installation Library
(DBCOLE.XDCZ1D.INSTALL.XDCXMIT)

This Member ...	... Produces this Installation Library
XDCADATA	DBCOLE.XDCZ1D.INSTALL.XDCADATA
XDCJCL	DBCOLE.XDCZ1D.INSTALL.XDCJCL
XDCLOAD	DBCOLE.XDCZ1D.INSTALL.XDCLOAD
XDCTEXT	DBCOLE.XDCZ1D.INSTALL.XDCTEXT

This Installation Library ...	... Has These Characteristics	
DBCOLE.---.XDCADATA	SPACE=(TRK,(1400,500,5))	DCB=(RECFM=VB,LRECL=8188)
DBCOLE.---.XDCJCL	SPACE=(TRK,(20,10,5))	DCB=(RECFM=FB,LRECL=80)
DBCOLE.---.XDCLOAD	SPACE=(TRK,(300,100,10))	DCB=(RECFM=U,BLKSIZE=32760)
DBCOLE.---.XDCTEXT	SPACE=(TRK,(10,50,50))	DCB=(RECFM=FB,LRECL=80)

Other Members of the Pre-Installation Library

Member	Purpose
$README	Contains a summary of the installation process.
$RECEIVE	Contains JCL for extracting the above installation libraries from the pre-installation library.

Figure 5 z/XDC Installation Libraries

To extract the installation libraries, run the JCL found in DBCOLE.XDCZ1D.INSTALL.XDCXMIT($RECEIVE). This job uses TSO's RECEIVE[4] commands (not SMP/E's RECEIVE command) to (re)build the four installation libraries shown in **Figure 5** on page 5.

Note, the z/XDC installation process assumes that all z/XDC related datasets have names that start with "DBCOLE.XDCZ1D.". You may change them to something else later (see "*Choose Naming Conventions ...*" on page 10 for more information about this); however for now, life for the both of us will be somewhat easier if you leave the datasets named as they are.

You are now ready to proceed with installing z/XDC.

INSTALLING THE EXTENDED DEBUGGING CONTROLLER

The installation process for z/XDC uses SMP/E[6]. There are essentially three phases to the installation process: The pre-SMP/E Phase, the SMP/E Phase, and the post-SMP/E Phase.

- In the pre-SMP/E Phase, installation datasets for z/XDC are created. These consist of the target datasets, the DLIB datasets, and the SMP datasets themselves. In addition, current maintenance for z/XDC has to be downloaded from our web site (www.colesoft.com).

- In the SMP/E Phase, SMP/E is executed to RECEIVE, APPLY, and ACCEPT the base z/XDC product into the z/XDC target and DLIB datasets. Also, current maintenance is RECEIVE'd and APPLY'd (but not ACCEPT'd) into the z/XDC target datasets.

- In the post-SMP/E Phase, a variety of tasks are performed in order to copy z/XDC elements from your target libraries to your production libraries, to activate z/XDC, and to make its various features available to your user community.

z/XDC z1.13's Compatibility with z/OS and with XDC's Prior Releases

z/XDC z1.13 is supported on z/OS r1.6 through at least R1.13.

Older releases of z/XDC are **not** supported on z/OS R1.13.

In general, multiple versions and releases of z/XDC can coexist on the same system. If you need to run this release of z/XDC concurrently with a prior version or release, review the following:

- **Installation Datasets:** For the SMP/E Phase of the installation process, you should create new SMP/E libraries and new DLIB and target z/XDC datasets that are separate from those used by prior versions and releases. (See "*Choose Naming Conventions ...*" on page 10.)

- **Load Module Names:** z/XDC load module names generally have not changed; therefore:

  - Two versions or releases of z/XDC libraries cannot effectively coexist in your linklist and LPA-list concatenations. Accordingly, during a "try-out" period, z/XDC's load modules need to not be installed into your linklist and LPA-list. Instead, they can be invoked via //JOBLIB, //STEPLIB, and equivalent techniques. (See "*Adding XDCPLPA*" on page 17 and "*Adding XDCLINK*" on page 17.)

  - Alternatively, all of z/XDC's load modules can be renamed. As distributed, all of z/XDC's load module names start with the characters **XDC**. You can renamed these modules to any other 3-character string (such as **Z1D**). If you do so, then multiple levels of z/XDC can exist in your linklist and LPA-list libraries; however, user programs that locate z/XDC by name may have to be reassembled. See "*Renaming z/XDC*" on page 48 for more information.

- **z/XDC's Service and Hook SVCs:** z/XDC internally installs and manages two SVC routines and several system intercepts. It does this in such a way that any two or more versions and releases of the SVCs can automatically coexist and not interfere with each other (regardless of whether or not

[6] **Now don't panic!** I myself am an SMP-phobe from way back! I feel your pain [:)]. Trust me, I've put a lot of effort into making this as painless a process as possible. If you don't like SMP, if you don't know SMP, if you don't **want** to know SMP, then this is the installation process for you. It is self contained, and it works very well. Just use the installation jobs and process that I've supplied, and you won't have to go anywhere near your system's SMP libraries. In fact, you will have an easier time then the experienced Sysprog who is probably going to insist on remangling this stuff into something that works "the right way". *[Oh well.]*

XDC has been renamed). See "*Installing z/XDC's Service and Hook SVCs*" on page 37 for more information.

- **Clists and ISPF Elements:** **Figure 10** on page 22 shows the old and new names of the various elements of z/XDC relating to ISPF. As you can see, the names of some of these elements have changed while others have not. Those elements whose names have changed can, of course, coexist with their older counterparts. Those elements whose names have not changed are backwards compatible with the older versions and releases of z/XDC. Accordingly, when installing the new z/XDC, you can replace the old clists and ISPF elements with the new ones. In fact you should perform these replacements because the older elements are not always forward compatible with the newer z/XDC.

- **z/XDC's Server Task:** The Server task ("Server/XDC") contains z/XDC's Cross Domain Facility (CDF) as well as other z/XDC support capabilities.

  z1.13's Server/XDC can be executed simultaneously with other copies of Server/XDC from prior versions and releases of z/XDC, even when the prior Server/XDC is part of a z/XDC that has not been renamed.

  In addition, multiple copies of CDF from the same release can be executed just so long as they are using z/XDC's having different names. (See "*Renaming z/XDC*" on page 48 for more information.)

  Each CDF, however, must be assigned to a unique VTAM node containing a uniquely named set of APPLID's. See "*Installing z/XDC's Cross Domain Facility (CDF)*" on page 43 for more information.

Installation Summary

The following is a summary of the required and optional steps for installing z/XDC.

Pre-SMP/E Phase:

1.) Choose a naming convention for z/XDC's datasets. (page 10)

2.) Download from our web site (www.colesoft.com) current maintenance for z/XDC. (page 11)

3.) `DBCOLE.XDCZ1D.INSTALL.XDCJCL` contains four SMP/E related jobs: `SMPCSI`, `SMPDDDEF`, `SMPBASE`, and `SMPMAINT`. Edit these jobs to meet local jobcard, dataset naming, and other requirements. (page 12)

4.) Run the `SMPCSI` job (from `DBCOLE.XDCZ1D.INSTALL.XDCJCL`) to create a CSI database and other SMP datasets for z/XDC. (page 12)

5.) Run the `SMPDDDEF` job (from `DBCOLE.XDCZ1D.INSTALL.XDCJCL`) to create DDDEFs in z/XDC's CSI database for z/XDC's target and DLIB datasets. (page 13)

SMP/E Phase:

1.) Run the `SMPBASE` job (from `DBCOLE.XDCZ1D.INSTALL.XDCJCL`) to RECEIVE, APPLY, and ACCEPT the z/XDC base product into the z/XDC target and DLIB datasets. (page 14)

2.) Run the SMPMAINT job (from DBCOLE.XDCZ1D.INSTALL.XDCJCL) to RECEIVE and APPLY current maintenance. **This step is REQUIRED! z/XDC will refuse to come up until maintenance has been APPLY'd!** (page 14)

Post-SMP/E Phase:

1.) The z/XDC target libraries can now be copied to or concatenated to various system libraries. (page 15)

2.) You may have to re-IPL your system, possibly with a CLPA (although a re-IPL usually is avoidable). (page 25)

3.) Licensing: Run z/XDC from a TSO session. The first time you do this, z/XDC automatically invokes a "License Definition" panel. You must fill in the required information in order to activate z/XDC for your system. (page 26)

4.) If your site uses a security system (RACF, CA-ACF2, or CA-Top Secret), then rules may have to be defined in that system to permit selected users to run z/XDC in various authorized modes. (page 28)

5.) An "XDCINIT" job needs to be setup to run at system IPL time to dynamically install a couple of SVCs and system intercepts that z/XDC needs for performing certain of its functions. (z/XDC's SVCs and intercepts must be dynamically reinstalled at every IPL.) (page 37)

6.) Perform an Installation Verification Test to verify that the installation of z/XDC to this point is correct. (page 39)

7.) Install the necessary PROCs, VTAM definitions, and system commands to enable use of z/XDC's Cross Domain Facility ("CDF") for debugging background jobs and system tasks. (page 43)

8.) The Installation Verification Test should now be repeated using CDF to verify that the facility is installed correctly. (page 46)

9.) If you want this release of z/XDC to coexist with prior versions or releases, then the most flexible way to accomplish that would be to change z/XDC's name to some other 3-character name (**Z1D** for example). (page 48)

10.) Set up the DBCOLE.XDCZ1D.XDCADATA library as a default MAPLIB[7] library for all z/XDC users. (page 50)

11.) z/XDC has a large number of user and installation settable profile options. You need to review the various default settings, make any changes you deem appropriate, and save the changed settings as a new default profile for other z/XDC users on your system. (page 51)

12.) z/XDC supports several user exits. They are "front/back end", "initialization", "termination", "resumption", "user-SVC", "user communications interface", and "user defined commands". Eventually, users may want to code one or more of these exit routines. Existing exits will have to be reassembled. (page 56)

13.) The XDCJES2 library contains an interface to z/XDC for debugging local mods to JES2. You may

[7] A **MAPLIB library** is a library that contains ADATA files created by an assembler and from which z/XDC's MAP and DMAP commands can build Source Image Maps for use in Source Level Debugging.

want to install this interface. (page 59)

Expanded discussions of the above outlined installation steps follow.

Pre-SMP/E Installation Phase

The installation process for z/XDC has been designed to use its own private target, DLIB, and SMP datasets, including its own private CSI ("Consolidated Software Inventory", a VSAM database for SMP/E). There are several reasons for this:

- The z/XDC product has no installation time dependencies on the release and maintenance levels of other products or of z/OS itself. (z/XDC runs on all supported versions of z/OS.) Accordingly, there is no need for it to be installed in the same target or DLIB zone as any other product or system.
- The SMP/E zone, options, and utilities definitions that work best for z/XDC's installation process may not be compatible with the definitions found within existing CSI databases for z/OS or other products.
- The z/XDC distribution contains several hundred macros and other elements, and ColeSoft cannot guarantee that the names of these elements are unique among all products.
- The private SMP datasets are small and pose no significant impact against free disk space.
- The use of private SMP datasets permits a uniform and simplified installation process for all our customers and, therefore, simplifies our task of providing as automated an installation and maintenance process as possible.
- The creation of the SMP datasets and the basic installation of the z/XDC product can be done with just four standard jobs requiring only minimal customization for local needs. **This permits even those programmers having little or no understanding of SMP/E to install z/XDC successfully.**
- Use of private SMP datasets makes it possible for individual user groups (where appropriate) to install their own private copies of the z/XDC product **and not have to involve unnecessarily a centralized Systems Programming staff** (except for certain elements of the Post-SMP/E Installation Phase).
- If severe problems occur during the SMP/E Phase, the current installation process can be abandoned and restarted simply by rerunning the four installation jobs (which will automatically delete and reallocate the SMP, DLIB, and target datasets).

Step 1: Choose Naming Conventions for z/XDC's Target, DLIB, and SMP Datasets

The dataset names for z/XDC's SMP libraries and for z/XDC's target and DLIB datasets all should include your favorite hi-level qualifier (mine is "DBCOLE"), the product name ("XDC"), and z/XDC's current release ("Z1D"). This will allow you to install separate releases of z/XDC into separate libraries, which is useful if you want to "try out" a new release before committing to it. In this Guide, I will use "DBCOLE.XDCZ1D." as the naming convention for the z/XDC datasets.

Step 2: Downloading Maintenance from our Web Site

Maintenance for release z1.13 of z/XDC is distributed in a cumulative maintenance file. Every distribution of z/XDC maintenance always contains all applicable maintenance that has been written from the time that z1.13 was released until the time that the maintenance file was created. Therefore, whenever you obtain a new maintenance file, you can always discard any older files that you may have previously obtained, regardless of whether or not you ever got around to APPLY'ing the older files.

z/XDC maintenance is provided in SMP/E format and in most cases is APPLY'd to z/XDC using the `SMPMAINT` job. See "*Step 2: Run the `SMPMAINT` Job*" on page 14 for details.

z/XDC maintenance is distributed from our Internet web site. Start at www.colesoft.com, and navigate as follows:

- From our home page, click on **Support**, then click on **Maintenance**. This will take you to our maintenance page.
- Read carefully and completely the entirety of the maintenance page. This is where last minute maintenance instructions and processing information is published. This is the only place where this information is published.
- Click on the "Contents and Dates" button, and read the information therein. This is where instructions specific to the particular maintenance file are published. This is the only place where this information is published.
- Select the appropriate file download button, and click on it in the manner instructed on the page. The file named *z1dmaint.ebc* will then be downloaded to your PC's hard drive.
- Upload *z1dmaint.ebc* from your PC to your mainframe into a sequential dataset named `DBCOLE.XDCZ1D.XDCMAINT` (depending upon the z/XDC dataset naming conventions that you have chosen). The upload type must be **binary** (ie, do not permit ASCII-to-EBCDIC translation). The DCB attributes of the target dataset need to be `RECFM=FB`, `LRECL=80`, and `BLKSIZE=n*80` (e.g. `BLKSIZE=27920`).

Alternatively, `DBCOLE.XDCZ1D.INSTALL.XDCJCL($FTPMAIN)` contains a job that can be used to download z/XDC z1.13's maintenance directly from www.colesoft.com to your mainframe. Just edit its JCL to meet your local requirements and then submit the job. When completed, a sequential dataset named `DBCOLE.XDCZ1D.XDCMAINT` will have been (re)created, and it will contain the most current z/XDC maintenance.

So, right now, please go to our web site and download current maintenance.

!!!!!!
PLEASE DO NOT CONTINUE THE z/XDC INSTALLATION PROCESS UNTIL THE MAINTENANCE HAS BEEN DOWNLOADED
(Otherwise, you will just have to repeat many of the installation steps later.)
!!!!!!

Now that you've downloaded the maintenance to your mainframe, you're ready to continue with the installation process. Don't apply the maintenance yet though, that's still a few steps down the road.

Step 3: Edit the DBCOLE.XDCZ1D.INSTALL.XDCJCL Jobs According to Local Requirements

"The z/XDC Product Distribution Package" on page 1 explained how to upload the DBCOLE.XDCZ1D.INSTALL.XDCJCL library to your mainframe from the Distribution Package.

The DBCOLE.XDCZ1D.INSTALL.XDCJCL library contains four jobs that should be used for installing z/XDC onto your system. These jobs are:

- **SMPCSI** creates SMP datasets (including a CSI) for z/XDC's private use. If one or more of the datasets already exist, then they are deleted and recreated. SMPCSI also initializes the CSI with appropriate zone definitions, options and utility entries, and DDDEFs for the other SMP datasets created.

- **SMPDDDEF** adds DDDEFs to the CSI for z/XDC's target and DLIB datasets. It also creates (or recreates) the target and DLIB datasets themselves.

- **SMPBASE** RECEIVE's, APPLY's, and ACCEPT's the z/XDC base product. It also APPLY's a maintenance stub.

- **SMPMAINT** first RESTORE's and REJECT's old maintenance (such as the maintenance stub) from the z/XDC target libraries. It then RECEIVE's and APPLY's new maintenance to the target libraries. This job should be used whenever you need to apply maintenance to z/XDC.

You may want to print listings of the four jobs for review. The jobs each contain helpful commentary that you should read. In particular, the commentaries describe the specific editing changes that you will have to make in order to customize the jobs to meet your local requirements.

Step 4: Run SMPCSI to Create SMP Datasets for z/XDC

SMPCSI creates SMP datasets (including a CSI database) for z/XDC's private use. (**Figure 6** on page 12 shows the datasets created.) If one or more of the datasets already exist, then they are deleted and recreated. SMPCSI also initializes the CSI database with appropriate zone definitions, options and utility entries, and DDDEFs for the other SMP datasets created.

DSNAME	Type	Comment
DBCOLE.XDCZ1D.CSI	VSAM	SMP/E's "Consolidated Software Inventory" database.
DBCOLE.XDCZ1D.SMPLOG	Sequential	SMP/E's activity log.
DBCOLE.XDCZ1D.SMPMTS	Partitioned	Temporary target level macro library.
DBCOLE.XDCZ1D.SMPPTS	Partitioned	Temporary sysmod storage library.
DBCOLE.XDCZ1D.SMPSCDS	Partitioned	Temporary backup storage library for target zone entries changed by JCLIN data during APPLY processing.
DBCOLE.XDCZ1D.SMPSTS	Partitioned	Temporary target level source module library.

Figure 6 The SMP/E Datasets Created by the SMPCSI Job

Note that if you have previously started the z/XDC installation process and you now wish to abandon that work and start over, you can do so simply by rerunning this SMPCSI job and the SMPDDDEF job (see below). These two jobs will delete and reallocate all of z/XDC's target, DLIB, and SMP datasets.

Read the commentary within the sample SMPCSI job for more information about required editing changes and expected completion codes.

Step 5: Run SMPDDDEF to Create z/XDC Target and DLIB Datasets

SMPDDDEF adds DDDEFs to the CSI database for z/XDC's target and DLIB datasets. It also creates the target and DLIB datasets themselves. (**Figure 7** on page 13 shows the datasets created.) If one or more of the datasets already exist, then they are deleted and recreated.

This job should be executed after the SMPCSI job. Read the commentary within the sample SMPDDDEF job for more information about required editing changes and expected completion codes.

DSNAME	Level	Comment
DBCOLE.XDCZ1D.XDCDLIB.XDCDLNKE	DLIB	All z/XDC load modules and program objects.
DBCOLE.XDCZ1D.XDCDLIB.XDCDTEXT	DLIB	All other z/XDC elements (Selected source, macros, panels, tables, parm files, etc.)
DBCOLE.XDCZ1D.XDCLINK	Target	z/XDC load modules for installation into a linklist PDS library.
DBCOLE.XDCZ1D.XDCLINKE	Target	z/XDC load modules for installation into a linklist PDSE library.
DBCOLE.XDCZ1D.XDCPLPA	Target	z/XDC load modules for installation into the PLPA.
DBCOLE.XDCZ1D.XDCADATA	Target	ADATA files including source image information for a large number of IBM system control blocks.
DBCOLE.XDCZ1D.XDCCMDS	Target	z/XDC command scripts for end-user use via z/XDC's **READ** command.
DBCOLE.XDCZ1D.XDCCDF	Target	A deprecated library containing installation elements for z/XDC's Cross Domain Facility. This library has been functionally replaced by DBCOLE.XDCZ1D.XDCSRVER. Customers migrating from z/XDC z1.10 or older releases should READ THE TOPIC **HELP WHATSNEW Z112 INCOMPATIBILITIES CDF** for important information.
DBCOLE.XDCZ1D.XDCMLIB	Target	ISPF message modules.
DBCOLE.XDCZ1D.XDCPLIB	Target	ISPF panels.
DBCOLE.XDCZ1D.XDCTLIB	Target	ISPF table modules.
DBCOLE.XDCZ1D.XDCCLIST	Target	Clists used for running z/XDC from ISPF.
DBCOLE.XDCZ1D.XDCSAMP	Target	Sample ACF2 security definitions, terminal LOGMODE tables and REXX execs.
DBCOLE.XDCZ1D.XDCSRVER	Target	A library containing a parameter file, VTAMLST data, and a sample PROC for running Server/XDC.
DBCOLE.XDCZ1D.XDCJES2	Target	Source code, an update file, and misc. JCL for installing a debugging interface into JES2.
DBCOLE.XDCZ1D.XDCASM	Target	Selected z/XDC sample source modules.
DBCOLE.XDCZ1D.XDCMACS	Target	Assembler macros needed for assembling the modules in XDCASM.
DBCOLE.XDCZ1D.XDCJCL	Target	Sample jobs for assembling the distributed source modules and for relinkediting the distributed load modules.
DBCOLE.XDCZ1D.XDCOBJ	n/a	An object file library for use when manually assembling the sample source modules in DBCOLE.XDCZ1D.XDCASM.

Note: DLIB libraries will never have z/XDC maintenance applied. Only target level libraries will be updated by z/XDC maintenance.

Figure 7 The Target and DLIB Datasets Created by the SMPDDDEF Job

SMP/E Installation Phase

This phase of z/XDC's installation procedure uses SMP/E to copy z/XDC elements into the several target libraries and the 2 DLIB libraries[8] (**Figure 7** on page 13). It also applies all current maintenance to z/XDC.

Step 1: Run the SMPBASE Job

The SMPBASE job RECEIVE's the z/XDC base product into SMP/E, APPLY's it to z/XDC's target libraries, and ACCEPT's it into the DLIB libraries.

The SMPBASE job can be executed directly after running the SMPDDDEF job. Read the commentary within the sample SMPBASE job for more information about required editing changes and expected completion codes.

In particular, please note that during the APPLY step, several warning messages may be issued by SMP/E. They are "GIM44402W SRCxxxxx WAS NOT ASSEMBLED ...". These messages can be ignored. They are for several source modules that are distributed only as samples. But these are the **only** warning messages that are expected. If other warnings occur (or if error messages occur) that you cannot resolve yourself, then please feel free to contact us for assistance.

Step 2: Run the SMPMAINT Job

APPLY'ing current maintenance is a REQUIRED step. z/XDC will refuse to come up until maintenance has been APPLY'd!

The SMPMAINT job must be executed to apply maintenance to z/XDC whenever you install z/XDC and again whenever you obtain a new maintenance file (*z1dmaint.ebc*).

Maintenance can be obtained only via a download from our Internet web site (www.colesoft.com). See *Step 2: Downloading Maintenance from our Web Site* on page 11 for more information.

If you have accumulated more than one *z1dmaint.ebc* maintenance file, always apply only the file having the latest date. All older files can be discarded. This is because z/XDC maintenance is always cumulative: Every maintenance file always contains all the maintenance necessary to bring z/XDC all the way from its distribution level up to the level that was current at the time that we built that file. In other words, every maintenance file always includes all of the maintenance previously included in all older files.

You must run the SMPMAINT job now in order to apply maintenance to z/XDC. If you have not yet downloaded maintenance from our web site, then go back to *Step 2* right now and do so.

The SMPMAINT job first uses SMP/E to RESTORE and REJECT all maintenance (including initial maintenance) that may have been previously APPLY'd. (SMP/E does this, of course, by copying product elements from the DLIB datasets to appropriate target level datasets. This is why it is important that z/XDC maintenance never be ACCEPT'd into the DLIB datasets.)

SMPMAINT then RECEIVE's and APPLY's the new maintenance into the target level datasets.

---

[8] "DLIB" is short for "distribution library", but that name does not accurately indicate what a DLIB library really is. These days, "DLIB libraries" are a set of libraries used by SMP/E as backups for target libraries (also sometimes called "TLIBs"). If bad program elements are installed onto target libraries, then theoretically they can be removed by copying corresponding elements from the DLIBs back to the target libraries. (This copying would be done by SMP/E's "RESTORE" command.)

During initial product installation, the SMPMAINT job must be executed directly after running the SMPBASE job. During subsequent maintenance processing, SMPMAINT can be executed by itself without the other three jobs.

Read the commentary within the sample SMPMAINT job for more information about required editing changes and expected completion codes.

Do not attempt to perform APPLY CHECK!

Normally, APPLY CHECK would be a sensible thing to do when APPLY'ing maintenance, and in the more typical case where products follow an incremental philosophy for their maintenance, this would work just fine. But it is not ever expected to work for our z/XDC product, and it should never be attempted.

This is because z/XDC maintenance is cumulative, not incremental. This means that each maintenance file that we develop contains all maintenance for the product, not just new maintenance. Consequently, before applying a new maintenance file, all old maintenance must first be removed.

So it follows that the maintenance files that we create are structured to APPLY successfully to a completely virgin, base copy of the product. They will not APPLY successfully to a copy of the product that already has older maintenance applied. Consequently, the **required** SMP/E command sequence must be a RESTORE followed by a REJECT followed by a RECEIVE followed by an APPLY. (This is what the SMPMAINT job does.)

If you tried to run an APPLY CHECK, it would check against a copy of the product for which RESTORE has not yet been done. In other words, the APPLY CHECK would be done against a level of the product for which the maintenance was not written. Consequently, it would always[9] fail. Bottom line:

- APPLY CHECK is not ever expected to be successful. Don't even try it.
- APPLY REDO is never expected to be successful. Don't even try it.
- ACCEPT MUST NEVER BE DONE!

Post-SMP/E Installation Phase

So much for the easy part! There still remains several tasks that need to be performed in order to complete the z/XDC installation process.

Step 1: Adding z/XDC's Target Libraries to Your System

z/XDC's various target libraries are shown in **Figure 7** on page 13. Of these, the following need to be "added" to your system libraries:

- DBCOLE.XDCZ1D.XDCLINK
- DBCOLE.XDCZ1D.XDCLINKE
- DBCOLE.XDCZ1D.XDCPLPA
- DBCOLE.XDCZ1D.XDCMLIB
- DBCOLE.XDCZ1D.XDCPLIB
- DBCOLE.XDCZ1D.XDCTLIB

[9] Actually, APPLY CHECK could succeed prior to the very first time you APPLY maintenance, but after that, it will always fail.

- DBCOLE.XDCZ1D.XDCCLIST
- DBCOLE.XDCZ1D.XDCCMDS

By "added" I mean that these libraries need to be either copied into or concatenated to your corresponding system libraries. The choice depends upon your local installation standards.

z/XDC's load modules are contained in:

- DBCOLE.XDCZ1D.XDCLINK
- DBCOLE.XDCZ1D.XDCLINKE
- DBCOLE.XDCZ1D.XDCPLPA.

Figure 8 on page 16 lists the individual load modules, what their functions are, and where they should go on your system. More specific information follows.

Module	System Library Function	Authorization Needed?	
XDC	an LPA-list library	No	Is the Extended Debugging Controller. Must reside in 24-bit storage.
XDCCALL	a linklist library	No	Is used to debug programs under TSO or in the batch.
XDCCALLA	a linklist library	Yes	Is used to debug authorized programs under TSO or in the batch.
XDCCDF	a linklist library	No	Provides z/XDC's Cross Domain Facility (CDF).
XDCCICSX	a linklist library	No	A support routine for debugging CICS transactions.
XDCCMD	a linklist library	No	Is used to debug commands under TSO. This is actually an alias for XDCCALL.
XDCCMDA	a linklist library	Yes	Is used to debug authorized commands under TSO. This is actually an alias for XDCCALLA.
XDCCXDC	a PDSE linklist library	No	A support routine for debugging C/C++ programs.
XDCEFMVS	a linklist library	No	REXX interface support routines.
XDCHELPM	a linklist library	No	Contains all of z/XDC's Online HELP.
XDCIVP	none	No	Is an "Installation Verification Program".
XDCLCNSE	a linklist library (PDS!)	No	Contains license control data.
XDCMAPS	a linklist library	No	Contains z/OS control block maps for use by the **DMAP** command.
XDCONLHG	none	No	Is a sample program that illustrates how to write a "User Communications Interface" exit for z/XDC.
XDCSERVE	a linklist library	Yes	Is Server/XDC.
XDCSTAMX	a linklist library	No	REXX interface support routines. This is actually an alias for XDCEFMVS.
XDCSYMED	a linklist library	No	Is an object deck SYM table editor.
XDCSYSIF	a linklist library	No	Contains the originating code from which z/XDC's various SVC routines and system intercepts are built.
XDCTFS	a linklist library	Yes	Provides Fullscreen Support for z/XDC.
XDCXITS	a linklist library	No	A sample user exit module that implements a **LIST TIOT** command. (See SRCXUCMD and SRCXITSR.)
XDC31	an LPA-list library	No	Is the portion of z/XDC that can reside in 31-bit storage.

Figure 8 z/XDC's Load Modules and Their Appropriate Target System Libraries

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 this library to understand

the discussion. If not, then please refer to IBM's **Initialization and Tuning Reference** manual[10] for more information, or consult with your systems programmer, as appropriate.

If necessary, you may also, of course, call us for assistance.

Adding XDCPLPA

DBCOLE.XDCZ1D.XDCPLPA contains load modules XDC and XDC31. "XDC" is z/XDC's primary load module. It must reside in 24-bit storage. "XDC31" contains that part of z/XDC that can reside in 31-bit storage.

These modules need to be placed into the system's PLPA. You can do so either by copying the modules to an existing PLPA library or by adding DBCOLE.XDCZ1D.XDCPLPA's dsname to the LPA library list.

If you already have a prior release of z/XDC installed in your PLPA, and if you need to install this new release for a "try-out" period without disturbing the older release, then you will have to rename z/XDC before you can do so. See *"Step 9: Renaming z/XDC"* (page 48) for details.

The XDC and XDC31 load modules will have to be reinstalled into the PLPA every time you apply z/XDC maintenance to the DBCOLE.XDCZ1D.XDCPLPA library.

If you want to add the DBCOLE.XDCZ1D.XDCPLPA library to the LPA-list, then you need to add DBCOLE.XDCZ1D.XDCPLPA's dsname to the LPA section of the active PROGxx member of a PARMLIB library.

You can use the SETPROG operator command to activate the modules immediately. The commands are:

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

Adding XDCLINK and XDCLINKE

Both DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCLINKE 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[11] libraries. XDCLINKE is a PDSE that contains program objects that must reside in PDSEs.

You may either copy the XDCLINK and XDCLINKE libraries to existing linklist libraries or add them to your linklist concatenation. Alternatively (but not recommended), you may leave the XDCLINK and XDCLINKE libraries out of the linklist entirely and, therefore, require users to access the z/XDC modules via //JOBLIB or //STEPLIB.

If you already have a prior release of z/XDC installed in your linklist, and if you need to install this new release for a "try-out" period without disturbing the older release, then you will have to rename z/XDC

[10] z/OS: SA22-7592.

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

before you can do so. See *"Step 9: Renaming z/XDC"* (page 48) for details.

If you copy the DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCLINKE modules into existing linklist libraries:

- The target linklist libraries need to be authorized.
- After copying the z/XDC load modules, perform an "LLA REFRESH" to make them usable (assuming the target linklist libraries did not expand into extra extents).
- The copying process may have caused your target linklist libraries to expand into extra extents. If this occurred, then you will have to either re-IPL or use the SETPROG operator command to rebuild the linklist. The following commands should work:

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

- This copying process will have to be repeated every time you apply z/XDC maintenance to the DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCLINKE libraries.

If you add the DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCLINKE libraries to the linklist concatenation:

- Add DBCOLE.XDCZ1D.XDCLINK's and DBCOLE.XDCZ1D.XDCLINKE's dsnames to the LNKLST section of the active PROGxx member of a PARMLIB library.
- The DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCLINKE libraries must be made authorized by your choice of the following methods:
  - Specify LNKAUTH=LNKLST in the active IEASYSxx member of a PARMLIB library.
  - Add DBCOLE.XDCZ1D.XDCLINK's and DBCOLE.XDCZ1D.XDCLINKE's dsnames and locations (volsers) to the APF section of the active PROGxx member of a PARMLIB library.
  - Both.
- Use the "SETPROG APF,---" and "SETPROG LNKLST,---" commands to authorize the DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCLINKE libraries and activate the linklist changes. Examples:

```
SETPROG APF,ADD,DSN=DBCOLE.XDCZ1D.XDCLINK,VOL=vvvvvv
SETPROG APF,ADD,DSN=DBCOLE.XDCZ1D.XDCLINKE,VOL=vvvvvv
SETPROG LNKLST,DEFINE,NAME=TEMP,COPYFROM=CURRENT,NOCHECK
SETPROG LNKLST,ADD,NAME=TEMP,DSN=DBCOLE.XDCZ1D.XDCLINKE,ATTOP
SETPROG LNKLST,ADD,NAME=TEMP,DSN=DBCOLE.XDCZ1D.XDCLINK,ATTOP
SETPROG LNKLST,ACTIVATE,NAME=TEMP
DISPLAY PROG,LNKLST,NAME=CURRENT
```

  These commands are documented in IBM's **z/OS: MVS System Commands**[12].

If you elect to leave the DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCLINKE libraries and modules out of the linklist entirely:

- The DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCLINKE libraries must be made authorized by adding their dsnames and locations (volser) to the APF section of the active PROGxx member of a PARMLIB library. You will have to 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

[12] z/OS: SA22-7627.

that if you 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 DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCPLPA libraries. This is because LIBDEF does not support authorized libraries.

XDCCALLA and XDCCMDA: Additional Considerations

XDCCALLA and XDCCMDA are authorized copies of the (nonauthorized) XDCCALL and XDCCMD load modules located in the DBCOLE.XDCZ1D.XDCLINK library. They allow users to debug authorized programs and command processors. These modules have been created with the "AC=1" attribute.

If you do not wish to permit authorized debugging, then you can create security system rules to prohibit it. See "*Defining z/XDC to Your Computer's Security System*" on page 28 for more information.

On the other hand, if you do want to permit authorized debugging via XDCCALLA and XDCCMDA, then you will have to tell TSO to let these two programs run authorized. (Otherwise, they will run non-authorized in TSO and, therefore, behave identically to XDCCALL and XDCCMD.) This is done by updating certain TSO tables.

To allow XDCCALLA and XDCCMDA to run authorized in TSO, you must add their names to the AUTHCMD parameter in the active IKJTSOxx member of a PARMLIB library. Then in order to activate the new AUTHCMD list, you can either re-IPL or use TSO's "PARMLIB" command to cause TSO to refresh its AUTHCMD list on the fly, without an IPL. For details, see **Figure 9** on page 20. For more complete information, see IBM's **Initialization and Tuning Reference** manual[13], the **TSO/E Customization** manual[14] and the **RACF Command Language Reference** manual[15].

[13] z/OS: SA22-7592.

[14] z/OS: SA22-7783.

[15] z/OS: SC22-7687.

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

```
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 option 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 the IKJTSOxx member of a PARMLIB libraries that you updated.

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 Option 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 9 Adding XDCCALLA and XDCCMDA to IKJTSOxx

Note that the above procedure is one of the steps needed to permit 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 XDCCALLA's name is in IKJTSOxx's AUTHCMD list, XDCCALLA will run authorized in the batch regardless. To control this use of XDCCALLA, you will have to define security system rules. See "*Defining z/XDC to Your Computer's Security System*" on page 28 for more information.

Adding XDCCLIST

The DBCOLE.XDCZ1D.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 z/XDC's Startup 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 DBCOLE.XDCZ1D.XDCCLIST, DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCPLIB, and

DBCOLE.XDCZ1D.XDCTLIB 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.
- (Recommended) 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 original library that contains the XDCCLIST clist.
- You may add the DBCOLE.XDCZ1D.XDCCLIST library to your existing clist library concatenations.

Although the XDCCLIST and XDCLBDEF clists have been revised from one release to the next, they remain compatible with older XDC's and z/XDC's. Accordingly, if you have an older XDC or z/XDC installed, then you can and should replace the older XDCCLIST and XDCLBDEF with these newer versions.

If you copy the DBCOLE.XDCZ1D.XDCCLIST library clists to an existing clist library, then be aware of the following:

- The DBCOLE.XDCZ1D.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 (option 3.3) to perform the copy. IEBCOPY cannot do the necessary record format conversion.
- This copying process will have to be repeated every time you apply z/XDC maintenance to the DBCOLE.XDCZ1D.XDCCLIST library.

If you add the DBCOLE.XDCZ1D.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 libraries having differing record formats.
- You must add a DD card for the DBCOLE.XDCZ1D.XDCCLIST library to the //SYSPROC concatenation of every logon PROC for every TSO user whom you expect to use z/XDC. (Note, systematic use of "// INCLUDE ---" JCL in your LOGON procs can ease this sort of burden considerably. See **z/OS MVS: JCL Reference**[16] for more information.)
- Alternatively, if you use logon clists (e.g. "PARM=%LOGON" in the TSO logon PROCs), you can recode that clist to add the DBCOLE.XDCZ1D.XDCCLIST library to the //SYSPROC concatenations.
- It does not matter where, in the //SYSPROC concatenation, the DBCOLE.XDCZ1D.XDCCLIST library occurs (first, last, middle) except that when an older XDC, XDC/SE, or z/XDC clist resides in the concatenation, then this newer library needs to be concatenated ahead of the older one.
- The first library in any concatenation either must have the largest BLKSIZE or must specify DCB= BLKSIZE= a value as large as or larger than the library having the largest BLKSIZE. The DBCOLE.XDCZ1D.XDCCLIST library's BLKSIZE is 27920.

[16] z/OS: SA22-7597.

Adding XDCMLIB, XDCPLIB, and XDCTLIB

The DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCPLIB, and DBCOLE.XDCZ1D.XDCTLIB libraries contain ISPF panels, messages, and table modules used for interfacing z/XDC to ISPF. (See **Figure 10** on page 22.) You may either copy these libraries to existing ISPF libraries, or you may add them to your existing library concatenations.

z1.13* Name	z1.12 Name	Type	Purpose
TFSXDC1D	TFSXDC1C	Table	A system-wide default profile for users.
TFSPROF	same	Table	An ISPF profile that causes ISPF to pass all PF-keys and ISPF commands through for processing by z/XDC.
TFSWIDE	same	Table	A system wide profile for use by users with large display terminals.
XDCZ1DA	XDCZ1CA	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.
XDCZ1DB	XDCZ1CB	Panel	A special purpose panel used in miscellaneous situations by z/XDC's interface to ISPF.
XDCPANEL	same	Panel	z/XDC's Startup Panel to be installed into ISPF so that user's can more easily invoke z/XDC.
XDCPHELP	same	Panel	Online Help for z/XDC's Startup Panel (XDCPANEL).
XDCPHLPx	same	Panel	Various additional Online Help for z/XDC's Startup Panel.
XDCM00	same	Message	Error messages used by the z/XDC Startup Panel.
XDCCLIST	same	Clist	Invoked internally by XDCPANEL to allocate datasets and then call XDCCALL (et.al.) to start the requested debugging session.
XDCLBDEF	same	Clist	Can be invoked by users to dynamically allocate z/XDC's ISPF libraries and display the XDCPANEL panel.

* **All** elements are backwards compatible with prior z/XDC releases.

Figure 10 z/XDC's ISPF Clists, Panels, Tables, and Message Modules

Alternatively, you may avoid adding the DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCPLIB, and DBCOLE.XDCZ1D.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 recommended.)

z/XDC's ISPF panels, tables, and message modules for release z1.13 have been updated with respect to those used by prior versions or releases. Those elements that are no longer compatible with older XDC's and z/XDC's have been renamed, while those that remain compatible retain their old names. Accordingly, if you have an older XDC or z/XDC installed, then you can and should replace the older panels, tables, and message modules with these newer versions, but also retain those older elements not replaced by newer elements.

If you copy DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCPLIB, and DBCOLE.XDCZ1D.XDCTLIB into existing ISPF libraries:

- This copying process will have to be repeated every time you apply z/XDC maintenance to the DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCPLIB, and DBCOLE.XDCZ1D.XDCTLIB libraries.

- Modify an invoking panel (such as ISR@PRIM or ISRUTIL) to invoke the XDCPANEL panel. (See "*Setting Up the XDCPANEL Panel or the XDCLBDEF Clist*" below on page 23 for details.)

If you add the DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCPLIB, and DBCOLE.XDCZ1D.XDCTLIB libraries to your ISPF library concatenations:

- You must add a DD card for the z/XDC libraries to the //ISPMLIB, //ISPPLIB, and //ISPTLIB concatenations of every logon PROC for every TSO user whom you expect to use z/XDC. (Note, systematic use of "// INCLUDE ---" JCL in your LOGON procs can ease this sort of burden considerably. See **z/OS MVS: JCL Reference**[17] for more information.)

- 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.

- It does not matter where, in the ISPF library concatenations, the z/XDC libraries occur (first, last, middle) except that when older XDC or z/XDC elements reside in the concatenations, then these newer libraries need to be concatenated ahead of the older ones.

- The first library in any concatenation either must have the largest BLKSIZE or must specify DCB= BLKSIZE= a value as large as or larger than the library having the largest BLKSIZE. The z/XDC library BLKSIZEs are all 27920.

- Modify an invoking panel (such as ISR@PRIM or ISRUTIL) to invoke the XDCPANEL panel. (See "*Setting Up the XDCPANEL Panel or the XDCLBDEF Clist*" below on page 23 for details.)

If (as recommended) you use the XDCLBDEF clist to dynamically allocate the DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCPLIB, DBCOLE.XDCZ1D.XDCTLIB, and DBCOLE.XDCZ1D.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 DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCPLIB, DBCOLE.XDCZ1D.XDCTLIB, and DBCOLE.XDCZ1D.XDCCLIST libraries.

- 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 DBCOLE.XDCZ1D.XDCCLIST library and not to issue an ALTLIB ACTIVATE command for it.

- Optionally, you may modify an invoking 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*" below on page 23 for details.) Alternatively, the user can simply run the XDCLBDEF clist from ISPF's option 6.

Setting Up the XDCPANEL Panel or the XDCLBDEF Clist

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 11** on page 24.

XDCPANEL can be invoked either directly ("PANEL(XDCPANEL)") or indirectly ("CMD(%XDCLBDEF)"). See "*Choosing Between Using XDCPANEL Directly or Indirectly via*

[17] z/OS: SA22-7597.

XDCLBDEF" on page 24 for important considerations.

In order to install XDCPANEL, see **Figure 11** on page 24 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". "X" is not recommended because that would conflict with ISPF's "exit" command.

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


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

Figure 11 ISPF Panel Mods for Invoking the XDCPANEL Panel

- A line needs to be added to the calling panel's &ZSEL=TRANS... command to invoke either the XDCPANEL panel or the XDCLBDEF clist, depending upon the considerations discussed below.

Choosing Between Installing XDCPANEL Directly or Indirectly via XDCLBDEF

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

- z/XDC's DBCOLE.XDCZ1D.XDCPLIB, DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCTLIB, and DBCOLE.XDCZ1D.XDCCLIST libraries must first be copied to or concatenated to you ISPPLIB, ISPMLIB, ISPTLIB, and SYSPROC libraries, as discussed above in "*Adding XDCMLIB, XDCPLIB, and XDCTLIB*" on page 22 and in "*Adding XDCCLIST*" on page 20.

- z/XDC's DBCOLE.XDCZ1D.XDCLINK, DBCOLE.XDCZ1D.XDCLINKE and DBCOLE.XDCZ1D.XDCPLPA libraries must first be copied to or concatenated to your STEPLIB, PLPA, and/or link-libraries, as discussed above in "*Adding XDCLINK and XDCLINKE*" on page 17 and "*Adding XDCPLPA*" on page 17.

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

- You need not copy or add the DBCOLE.XDCZ1D.XDCPLIB, DBCOLE.XDCZ1D.XDCMLIB, DBCOLE.XDCZ1D.XDCTLIB, and DBCOLE.XDCZ1D.XDCCLIST libraries to your ISPF libraries. They will be dynamically allocated by XDCLBDEF.

- But you still have to copy the XDCLBDEF member of the DBCOLE.XDCZ1D.XDCCLIST library to a SYSPROC library.

- For **non**authorized debugging, you need not add the DBCOLE.XDCZ1D.XDCLINK, DBCOLE.XDCZ1D.XDCLINKE and DBCOLE.XDCZ1D.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 DBCOLE.XDCZ1D.XDCLINK, DBCOLE.XDCZ1D.XDCLINKE and DBCOLE.XDCZ1D.XDCPLPA 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 XDCLINK, XDCLINKE and XDCPLPA libraries into your STEPLIB, PLPA, and/or link-libraries (as discussed in "*Adding XDCLINK and XDCLINKE*" on page 17 and "*Adding XDCPLPA*" on page 17).

- Use of the XDCLBDEF clist makes it easier to offer multiple versions or releases of z/XDC to your ISPF users. Users can run the XDCLBDEF clist from ISPF's option 6.

- Alternatively, you can setup two options in an invoking ISPF panel: "D,'PANEL(XDCPANEL)'" to invoke the old production version or release of z/XDC and "DNEW,'CMD(%XDCLBDEF)'" to invoke the new z/XDC release during a "try-out" period.

Adding XDCCMDS

The DBCOLE.XDCZ1D.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 "*Defining z/XDC to Your Computer's Security System*" on page 28), be sure to give your users "read" access to this library.

Step 2: Re-IPL (Maybe)

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

- When you added the DBCOLE.XDCZ1D.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 re-IPL.

- If you added the DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCLINKE libraries to the linklist, did you also issue the following commands?
```
SETPROG APF,ADD,DSN=DBCOLE.XDCZ1D.XDCLINK,VOL=vvvvvv
SETPROG APF,ADD,DSN=DBCOLE.XDCZ1D.XDCLINKE,VOL=vvvvvv
SETPROG LNKLST,DEFINE,NAME=TEMP,COPYFROM=CURRENT,NOCHECK
SETPROG LNKLST,ADD,NAME=TEMP,DSN=DBCOLE.XDCZ1D.XDCLINKE,ATTOP
SETPROG LNKLST,ADD,NAME=TEMP,DSN=DBCOLE.XDCZ1D.XDCLINK,ATTOP
SETPROG LNKLST,ACTIVATE,NAME=TEMP
```
If not, then do it now or re-IPL.

- 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 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 re-IPL.

- If you added the DBCOLE.XDCZ1D.XDCLINK and/or DBCOLE.XDCZ1D.XDCLINKE and/or DBCOLE.XDCZ1D.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=DBCOLE.XDCZ1D.XDCLINK,VOL=vvvvvv
SETPROG APF,ADD,DSN=DBCOLE.XDCZ1D.XDCLINKE,VOL=vvvvvv
SETPROG APF,ADD,DSN=DBCOLE.XDCZ1D.XDCPLPA,VOL=vvvvvv
```

  If not, then do it now or re-IPL.

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

```
PARMLIB UPDATE(xx)
```

  If not, then do it now or re-IPL.

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 process is entered by z/XDC automatically the first time that you try to use z/XDC. So in order to define your license to z/XDC, you must run z/XDC.

At this stage of the installation process you can run z/XDC as follows:

- The License Definition process is going to cause z/XDC to rewrite to the XDCLCNSE load module on disk. Consequently, you must insure that your TSO userid has write access to the library containing that load module.
- It is because of this license rewrite that the XDCLCNSE load module must reside in a PDS, not a PDSE. z/XDC does not have the technology needed to rewrite into a PDSE.
- Logon to a TSO terminal and proceed either to TSO's READY prompt or to ISPF's option 6 panel.
- In order to run z/XDC, XDCCALL is going to need access to the XDC, XDC31, XDCTFS, and XDCLCNSE load modules. If any of the libraries containing these modules are not in TSO's or ISPF's search order, then use TSO's ALLOC command to allocate them to ddname TASKLIB. Example:

```
ALLOC FILE(TASKLIB) REUSE SHR +
      DA('DBCOLE.XDCZ1D.XDCPLPA' +
         'DBCOLE.XDCZ1D.XDCLINK')
```

- From TSO READY or from ISPF option 6, start z/XDC as follows:
  - If the XDCCALL load module has already been installed into TSO's load module search order (linklist library, STEPLIB library, or ISPLLIB library), then type:

```
XDCCALLA IEFBR14
```

  - Otherwise, type:

```
CALL 'DBCOLE.XDCZ1D.XDCLINK(XDCCALLA)' 'IEFBR14'
```

- z/XDC may display its System Interface installation Report (DBC514I messages). If it does, then the report can be disregarded at this time. Just press ENTER until a screen is displayed that is similar to the one shown in **Figure 12** on page 27.

```
TCB#9 RB#1 ---------------- z/XDC ISPF INTERFACE -----------------------------
XDC ===>
   Press PF3 to vcheck changes and exit if valid or remain if not. Press PF4 to
   discard changes and exit.

DBC515E XDC HAS NOT BEEN LICENSED TO RUN WITHIN THE CURRENT SYSPLEX (nnnnnnnn)
   LICENSING CONTROL DATA ALTER/DISPLAY FOR z/XDC RELEASE z1.13

   Current CPU:    xxxx--xxxxx*  Current SYSPLEX:  nnnnnnnn*
   License Dsname: DBCOLE.XDCZ1D.XDCLINK(XDCLCNSE)

   Customer ID:           License Start Date: mm/dd/yyyy  base/XDC Quota: ----
   Control Key: 00000000 License End   Date: mm/dd/yyyy  auth/XDC Quota: ----
                         Expiration Warning: 037 (Days)  asm/XDC  Quota: ----
                                                          c/XDC    Quota: ----

       CAP   TYPE  SER#   FLAG  SYSPLEX      CAP   TYPE  SER#   FLAG  SYSPLEX
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
       ____  ____  _____  ____  ________     ____  ____  _____  ____  ________
```

* These fields display the machine type, CPU serial number, and SYSPLEX name of the **current computer** on which z/XDC is running.

Figure 12 z/XDC's License Definition Screen

- Included in your shipment is a cover letter that shows the Licensing Control Data (LCD) that you need to fill into the License Definition Panel. Once you have typed the correct LCD, you may press **PF3** to cause z/XDC to accept or reject the data. If the data is accepted, z/XDC will automatically write it to disk (into the XDCLCNSE load module) and proceed to begin a normal debugging session.

- If for some reason, z/XDC refuses to accept the LCD that you supply, then you can press **PF4** to abort the licensing process. You then should contact ColeSoft to resolve the problem.

- Once you have successfully entered the correct Licensing Control Data, you may experiment with z/XDC, or you may terminate the debugging session by pressing **PF3** or **PF4**.

- If the XDCLCNSE load module is located in the linklist, then you will have to perform an "LLA REFRESH" to cause MVS to refresh its cached copy of the load module. If you don't, then subsequent attempts to use z/XDC may behave as if the licensing process that you have just now completed never took place. To avoid this problem, you must issue the following system operator

command: `F LLA,REFRESH`

- Similarly, if your data center uses a caching product from Computer Associates called "PMO" with "QFETCH", and if the load library containing the XDCLCNSE module is cached by that product, then you will have to issue the following command to cause that product to refresh its image of the load library: `F PMO,D=libdsn`

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 again, you can do so by starting a normal z/XDC debugging session (such as "XDCCALL IEFBR14") and then issuing the **LICENSE** command.

After you have finished the license definition process, you should then IEBCOPY the XDCLCNSE load module from your production library back to your SMP/E target library (`DBCOLE.XDCZ1D.XDCLINK`) and to your SMP/E DLIB library (`DBCOLE.XDCZ1D.XDCDLIB.XDCDLINK`). Doing this will avoid problems in the future when you apply new maintenance and then decide to mass-copy all of z/XDC's load modules from the target library over to your production libraries.

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.

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 integrity with z/XDC.

There are, however, certain circumstances under which an installation may wish to restrict use of z/XDC to authorized personnel only:

1.) When a debugging session is started via the XDCCALLA or XDCCMDA program, or when a z/XDC interface is installed into an authorized program, z/XDC will receive control in an authorized mode. When this happens, the user of z/XDC can zap arbitrary system storage. Note that in a properly secured computer center, only appropriate personnel will have the capability of installing a z/XDC interface into an authorized program.

2.) 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. 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.

3.) When a user logs onto z/XDC's Cross Domain Facility, system security is called to verify that user's identity and again to determine which background jobs or tasks he is permitted to connect to for debugging.

If you need to control these kinds of accesses, then you will need to install certain z/XDC defined access control rules into your security system.

For More Information About z/XDC Security ...

The following pages describe what kinds of security restrictions can be set up for z/XDC users and how to establish those restrictions. There also is available 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 z1.13 User's Guide** or z/XDC's Online Help. The title of the discussion is named "Help Security". To read it, start a z/XDC debugging session[18], and then type the **HELP SECURITY** command.

z/XDC's Standard Security Method

z/XDC issues calls to the installation's own security system via IBM's "System Authorization Facility" (i.e. the "SAF interface", a standard facility in all z/OS systems, secured or otherwise). 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:

1.) Upon the first use within a debugging session of z/XDC in an authorized mode, z/XDC checks for "READ" access to a z/XDC defined resource named "XDC.AUTH".

2.) 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[19] 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 call is made for each different address space so accessed.

3.) The first time that 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" resource.

4.) Any time that a storage altering command is used to alter any storage anywhere, z/XDC checks for "UPDATE" access to a resource named "XDC.ZAP.---" (where "---" 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. (For more information, see **HELP SECURITY ZAP** either in z/XDC's Online Help or in **z/XDC z1.13 User's Guide**.)

5.) When a user signs onto the Cross Domain Facility from an idle VTAM terminal, CDF calls security to verify that user's TSO id and password. (When a user signs onto CDF 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.)

6.) When a user who has signed onto CDF selects a background job or task to debug, if that job's ownerid does not match the user's userid, then CDF 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" resource.

Important: For more detailed descriptions of z/XDC's resource access control rules, see the **HELP SECURITY** topic in **z/XDC z1.13 User's Guide** or in z/XDC's Online Help.

---

[18] From TSO's READY prompt or from ISPF's Option =6, type "XDCCALL IEFBR14".

[19] 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.

Using z/XDC Prior to Creating Security Definitions

z/XDC's security interface explicitly supports three security systems: RACF, CA-ACF2, and CA-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 CA-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). Accordingly, in RACF and CA-Top Secret shops, z/XDC can be installed and testing during a trial period without having to setup security definitions[20]. But security concerned installations should make the necessary definitions prior to z/XDC's being released to the user community for general use.

CA-ACF2 is an exception. It "disallows" (RC=8 from the RACROUTE macro) accesses to unprotected resources; therefore, in CA-ACF2 shops suitable security definitions for z/XDC will have to be made before z/XDC can be used fully.

Using z/XDC in RACF Protected Systems

Figure 13 on page 31 shows the RACROUTE macro operands used by z/XDC in RACF protected systems. Note that z/XDC uses the resource class named "FACILITY". This is an IBM defined "catch-all" class used to control accesses to miscellaneous system facilities. IBM uses this class for controlling storage dumps, access to vector processors, logon passwords for JES2/3 RJE/RJP and NJE connections, and miscellaneous facilities in various program products. z/XDC's use of this class avoids the necessity of requiring the customer to create or modify a "user" Class Descriptor Table.

[20] 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 User's Guide or the Online Help.

Call Type	ATTR=	ENTITY Value***	Resource Class
Authorized mode	READ	`CL44'XDC.AUTH'`	FACILITY
Read a foreign address space	READ	`CL44'XDC.FASM.asname'`*	FACILITY
Zap a foreign address space**	UPDATE	`CL44'XDC.FASM.asname'`*	FACILITY
Debug a program via CDF	UPDATE	`CL44'XDC.FASM.asname'`*	FACILITY
Zap storage**	UPDATE	`CL44'XDC.ZAP.---'`	FACILITY
(See HELP SECURITY LOSTLOCKS)	READ	`CL44'XDC.LOSTLOCKS'`	FACILITY

*"Asname" is replaced by the 8-character name of the address space that the user is trying to access.

**Zaps that require permission under the "XDC.FASM.asname" rule, still also require permission under an appropriate "XDC.ZAP.---" rule.

***All entity values start with "XDC" regardless of z/XDC's name. (See "*Renaming z/XDC*" on page 48 for more information.)

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

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

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

Figure 13 RACROUTE Macro Operands Used by z/XDC in **RACF** Protected Systems

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

- Use RACF's RDEFINE command (as follows) to create in the FACILITY class default profiles prohibiting access to all of z/XDC's authorized facilities for all users:

  ```
  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 "*Renaming z/XDC*" on page 48 for more information.)

- Depending on your specific circumstances, you may have to issue one or more of the following commands to properly activate the FACILITY class:

  ```
  SETROPTS CLASSACT(FACILITY)
  ```

  If this class has not previously been activated on your system.

```
SETROPTS GENERIC(FACILITY)
```
Because "XDC.FASM.*" and "XDC.ZAP.*" are generic profiles.

```
SETROPTS RACLIST(FACILITY)
```
To boost RACF performance when scanning generic profiles.

- Use RACF's "PERMIT" command to give the XDCSRVER job READ access to the XDC.AUTH rule:
  ```
  PERMIT XDC.AUTH CLASS(FACILITY) ID(ownerid) ACCESS(READ)
  ```
  Where "ownerid" is the RACF ownerid under which the XDCSRVER job will execute.

- Use RACF's "PERMIT" command and additional "RDEFINE" commands to grant individual users and user groups accesses to z/XDC's various authorized facilities. Example: The following commands permit user DBCOLE to use z/XDC's Foreign Address Space Mode to display but not to zap JES2's private areas.
  ```
  RDEFINE FACILITY XDC.FASM.JES2 UACC(NONE)
  PERMIT XDC.FASM.JES2 CLASS(FACILITY) ID(DBCOLE) ACCESS(READ)
  ```

- Use the SETROPTS command to refresh the changes made to the facility class:
  ```
  SETROPTS RACLIST(FACILITY) REFRESH
  RLIST FACILITY *
  ```

For additional information, please refer to RACF's **Security Administrator's Guide**[21] and to its **Command Language Reference**[22].

Also, for a comprehensive discussion about creating the XDC.AUTH, XDC.ZAP.---, and XDC.FASM.--- rules and about how z/XDC uses them, see the **HELP SECURITY** topic in **z/XDC z1.13 User's Guide** or in z/XDC's Online Help.

Using z/XDC in CA-ACF2 (Release 6.4 and Newer) Protected Systems

Figure 14 on page 33 shows the RACROUTE macro operands used by z/XDC in CA-ACF2 protected systems. To implement z/XDC security, several control definitions need to be made within CA-ACF2:

- A GSO CLASMAP.XDC record may need to be defined to set the resource type to be used for the validations.
- A "D-RXDC" entry needs to be added to the INFODIR control record.
- A GSO SAFDEF.XDC record may need to be defined to ensure the AUTH request is validated.
- Suitable "generalized resource rules" with TYPE(XDC) need to be created.[23]
- An "xxxSRVER"[24] logonid needs to be created for the Cross Domain Facility. This id must be assigned at least the following attributes: MUSASS, STC, and RESTRICT.

[21] z/OS: SA22-7683.

[22] z/OS: SC22-7687.

[23] The name of the class, the SAFDEF record, 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 above in "*Step 9: Renaming z/XDC*" on page 48).

[24] Where "xxx" matches z/XDC's name.

Call Type	ATTR=	ENTITY Value	CLASS, REQSTOR, and SUBSYS Values***
Authorized mode	READ	`CL44'AUTH'`	XDC
Read a foreign address space	READ	`CL44'FASM.asname'`*	XDC
Zap a foreign address space**	UPDATE	`CL44'FASM.asname'`*	XDC
Debug a program via CDF	UPDATE	`CL44'FASM.asname'`*	XDC
Zap storage**	UPDATE	`CL44'ZAP.---'`	XDC
(See HELP SECURITY LOSTLOCKS)	READ	`CL44'LOSTLOCKS'`	XDC

*"Asname" is replaced by the 8-character name of the address space that the user is trying to access.

**Zaps that require permission under the "XDC.FASM.asname" rule, still also require permission under an appropriate "XDC.ZAP.---" rule.

***The CLASS, REQSTOR, and SUBSYS value of "XDC" must be used for security calls regardless of whether or not you have renamed z/XDC. (See "*Renaming z/XDC*" on page 48 for more information.)

CA-ACF2 Example: An attempt by a user to zap JES2's address space would cause z/XDC to issue the following RACROUTE macro:

```
          RACROUTE REQUEST=AUTH,ATTR=UPDATE,REQSTOR=XDC,
                SUBSYS=XDC,CLASS=CLASS-1,ENTITY=RESOURCE

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

Figure 14 RACROUTE Macro Operands Used by z/XDC in **CA-ACF2** Protected Systems

More specifically, a security officer needs to modify the CA-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 SAFDEF.XDC 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 CA-ACF2 commands necessary to "refresh" the control records you have modified (SAFDEF, CLASMAP, and INFODIR). This will activate the changes you have made.

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

- From TSO, type "ACF", then type "SET RESOURCE(XDC)".
- Use CA-ACF2's COMPILE and other commands to created 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)`
  Controls use of z/XDC in authorized mode.

  `$KEY(ZAP.descriptions) TYPE(XDC)`
  Controls who can zap what storage. See the HELP SECURITY topic in **z/XDC z1.13 User's Guide** or in z/XDC's Online Help for details concerning "descriptions".

  `$KEY(FASM.asname) TYPE(XDC)`
  `UID(...) SERVICE(READ)`
  Controls who can have "read" access to the foreign address space named "asname".

  `$KEY(FASM.asname) TYPE(XDC)`
  `UID(...) SERVICE(UPDATE)`
  Controls who can have "zap" access and CDF access to the foreign address space named "asname".

- CA-ACF2 permits $KEY values to contain "wildcard" characters (asterisks), thus it is possible, for example, to create one "FASM.asname" rule to control accesses to multiple address spaces. (CA-ACF2 refers to the use of wildcards as "masking"). Remember however, if such generic 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).

CA-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 xxxCALL, xxxCMD, xxxCALLA, xxxCMDA, xxxTFS, and xxx, where **xxx** match's z/XDC's name (usually **XDC**). (See "*Renaming z/XDC*" on page 48 for more information.)

Finally, in order to run the Cross Domain Facility subserver task (see "*Installing z/XDC's Cross Domain Facility (CDF)*" on page 43), 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 "xxxSRVER"[25], and that name must match the name of the PROC used to start Server/XDC. You also must give the xxxSRVER logonid READ access to the AUTH rule.

See CA-ACF2's **Administrator's Guide** for more information about defining generalized resource rules, about logonids, and about modifying the control records. Also, member ACF2SAMP in `DBCOLE.XDCZ1D.XDCSAMP` contains several examples of various z/XDC type generalized resource rules.

Also, for a comprehensive discussion about creating the AUTH, ZAP.---, and FASM.--- rules and about how z/XDC uses them, see the HELP SECURITY topic in **z/XDC z1.13 User's Guide** or in z/XDC's Online Help.

[25] Where "xxx" matches z/XDC's name (usually **XDC**). See "*Renaming z/XDC*" on page 48 for more information.

Using z/XDC in CA-Top Secret Protected Systems

Figure 15 on page 35 shows the RACROUTE macro operands used by z/XDC in CA-Top Secret protected systems. Note that 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 ("Top Secret Security") 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,TENV[26],MULTIUSER,NONPWR,NOTSOC)
```

Call Type	ATTR=	ENTITY Value	Resource Class***
Authorized mode	READ	CL44'AUTH'	XDC
Read a foreign address space	READ	CL44'FASM.asname'*	XDC
Zap a foreign address space**	UPDATE	CL44'FASM.asname'*	XDC
Debug a program via CDF	UPDATE	CL44'FASM.asname'*	XDC
Zap storage**	UPDATE	CL44'ZAP.---'	XDC
(See HELP SECURITY LOSTLOCKS)	READ	CL44'LOSTLOCKS'	XDC

*"Asname" is replaced by the 8-character name of the address space that the user is trying to access.

**Zaps that require permission under the "XDC.FASM.asname" rule, still also require permission under an appropriate "XDC.ZAP.---" rule.

***The Resource Class of "XDC" is used for security calls regardless of z/XDC's name. (See "*Renaming z/XDC*" on page 48 for more information.)

For CA-Top Secret protected systems, z/XDC issues the RACROUTE macro with "REQUEST=AUTH" and with REQSTOR and SUBSYS not set. Example: An attempt by a user to zap JES2's address space would cause z/XDC to issue the following RACROUTE macro:

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

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

Figure 15 RACROUTE Macro Operands Used by z/XDC in **CA-Top Secret** Protected Systems

- Create an ACID ("access id") named "XDCCDF". This will be the ACID assigned to the Cross

[26] Support for "TENV" has been dropped in release 4.3 of TSS. Therefore, this operand should be omitted if you are running a 4.3 or newer release of TSS.

Domain Facility's startup proc:

```
TSS CREATE(XDCCDF) NAME('name') DEPT(ownerid) PASS(NOPW) FAC(STC,TSO)
    MASTFAC(XDCCDF)
```

- Assign xxxSRVER's[27] startup proc to use the XDCCDF ACID:

```
TSS ADD(STC) PROC(xxxSRVER)[28] ACID(XDCCDF)
```

- Allow users to logon to xxxCDF:

```
TSS ADD(userid) FAC(XDCCDF)
```

- Define a resource named "XDC" in the RDT ("Resource Definition Table"). "RESCODE" may specify any previously unused 2-digit code:

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

  Note, starting with R4.4, CA-Top Secret has been shipping with the "XDC" resource class predefined. Unfortunately, at first they got it wrong! Among other things, they assigned the z/XDC resource class an attribute of "SHORT", while z/XDC actually requires that the class's attribute be "LONG".

  Anyway, Computer Associates eventually discovered their errors, and so they developed the following fixes. These TSS fixes are required for using z/XDC in a CA-Top Secret environment:

  - PTF GO64801 (available on the 9506 maintenance tape) replaces module TSSRTAB. It corrects the z/XDC class's length attribute from SHORT to LONG.
  - APAR BB15253 corrects an "invalid keyword" error that occurs with the "TSS ADD(ownerid) XDC(rulename)" command.

  Presumably, R5.0 and newer releases of CA-Top Secret come with the "XDC" resource predefined correctly. To summarize:

  - If you are using a release of CA-Top Secret that is older than R4.4, then you have to issue the "TSS ADD" command shown above.
  - If you are using R4.4 or any newer release of CA-Top Secret, then you do not have to issue the "TSS ADD" command.
  - But if you are experiencing problems with installing z/XDC's definitions into TSS, then you will have to apply the TSS maintenance described above.

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

```
TSS ADD(ownerid) XDC(AUTH)
TSS ADD(ownerid) XDC(ZAP.)
TSS ADD(ownerid) XDC(FASM.)
```

  If these commands fail with an "INVALID KEYWORD" message, then apply the TSS maintenance described above, and then try again.

- 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 access to JES2's private areas:

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

---

[27] Where "xxx" matches z/XDC's name.

[28] In TSS release 4.3, and in TSS release 4.2 with fix #552 applied, the program executed by the xxxSRVER PROC will be able to use the XDCCDF "facility" regardless of whether or not the name of the program invoked by the PROC matches the "PGM=XDCTFS" parameter in the "facility" definition. In TSS release 4.2 systems with fix #552 not applied, then the program executed by the xxxCDF PROC must match the "PGM=XDCSERVE" parameter in the "facility" definition.

Note that with CA-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.

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

Also, for a comprehensive discussion about creating the AUTH, ZAP.---, and FASM.--- rules and about how z/XDC uses them, see the HELP SECURITY topic in **z/XDC z1.13 User's Guide** or in z/XDC's Online Help.

Another Approach to z/XDC Security

If you are installing z/XDC for the exclusive use of non-authorized users, and if you are not interested in using the Cross Domain Facility for debugging background jobs, then you might consider a less complicated approach to z/XDC security. Simply place all of z/XDC's load modules into a non-authorized JOBLIB or STEPLIB library (or linklist library as long as LNKAUTH=APFTAB is specified in the active `IEASYSxx` member of the `PARMLIB` libraries). This will make it impossible for z/XDC to be invoked by an authorized program. In addition you may want to use normal system security dataset access rules to control who among the non-authorized users may or may not use z/XDC.

However, in order to install z/XDC's service SVCs (see "*Installing z/XDC's Service and Hook SVC ...*" below on page 37), you will still have to make a second copy of the z/XDC load modules and place them into an authorized library for use by the XDCINITc job (see below). Just be sure to use system security dataset access rules to make the authorized library non-executable by all users except the XDCINITc job.

Step 5: Installing z/XDC's Service and Hook SVCs (via the XDCINITc Job)

In order to make the full range of z/XDC's facilities available for use, two special SVCs and several system intercepts must be installed for z/XDC. These SVCs provide various services that help improve the power of z/XDC. The SVCs 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. They cannot be misused by **un**authorized callers to subvert system security or integrity.

z/XDC's SVCs need not, and in fact cannot, be installed by the normal methods. Instead, they must be dynamically installed after every IPL by a special XDCINITc PROC that must be executed at or after IPL time. The XDCINITc PROC is shown in **Figure 16** on page 38. A copy of it can be found in `DBCOLE.XDCZ1D.XDCJCL`.

. This PROC must be placed into a system PROC-library (such as `SYS1.PROCLIB`).

The xxxINITc Proc

```
//*
//xxxINITc EXEC PGM=xxxCALLA,PARM=IEFBR14,TIME=1
```

Place the following command into a COMMNDxx member of a PARMLIB Library:

```
COM='S xxxINITc'
```

Where **xxx** matches z/XDC's name (usually "XDC"), and where "c" can be any alphanumeric (or omitted) character.

Figure 16 XDCINITc PROC and Associated COMMNDxx Command

Normally, the PROC must be named "XDCINIT" or "XDCINITc", where "c" can be any alphanumeric character. (See "*What the XDCINITc Job Does*" below on page 38 for why.) However, if you have renamed z/XDC to Z1D, DBC, or some other name (see "*Renaming z/XDC*" on page 48), then both the name of the XDCINITc PROC and the reference within the PROC to "XDCCALLA" must also be similarly renamed.

In order to insure that XDCINITc is executed on every IPL, you should add the command shown in **Figure 16** on page 38 to the active COMMNDxx member of the PARMLIB libraries. Again, if z/XDC has been renamed, then the command's reference to XDCINITc should be similarly changed. See IBM's **Initialization and Tuning Reference** manual[29] for detailed discussions of the PARMLIB libraries.

What the XDCINITc Job Does

The XDCINITc job does little more than invoke z/XDC in an authorized mode. Whenever z/XDC runs authorized, it checks to see if its service SVC, its hook SVC, and its several system intercepts are installed. If any elements are either not installed or are at a maintenance level that is older than XDC itself, then those elements are either installed or reinstalled.

In the case of the two SVCs, z/XDC automatically selects two available SVC numbers (199 and 198 or lower[30]), and it assigns them to the SVCs. (Usually, the hook SVC will be assigned to SVC #199, and the service SVC will be assigned to SVC #198, but this is not guaranteed.)

When (re)installation of the system interface is completed, z/XDC issues a detailed System Interface Initialization Report (messages DBC514I) showing exactly what it was and was not able to do, and why.

During the SVC installation process, z/XDC also does the following:

- It builds for itself a subsystem control table (SSCT) named "XDCc" (where "c" is a release specific special character) and uses it, among other things, as a place to save its SVC numbers.
- It installs various other system routines to support deferred breakpoints (see the **HELP BREAKPOINTS DEFERRED** topic in **z/XDC z1.13 User's Guide** or in z/XDC's Online Help)

[29] z/OS: SA22-7592.

[30] SVC numbers lower than 200 are chosen so as not to interfere with other user SVCs.

and certain end-of-task and end-of-memory cleanup functions.

If z/XDC is invoked in a job or system task whose name is "XDCINITc", then after insuring that its service and hook SVCs are installed, it terminates without entering into a debugging session. Thus, it is important that the XDCINITc job that is run at IPL time be named "XDCINITc" (or "xxxINITc" if z/XDC has been renamed to xxx). Otherwise, a spurious interactive debugging session will be started at the operator's console at IPL time (generally **not** a desirable occurrence).

Running XDCINITc Jobs for Multiple Versions and Releases of XDC

If you are installing this z1.13 release of z/XDC to run in parallel with an older version or release of z/XDC, then you will have to have two XDCINIT jobs on your system, one for installing z1.13's SVCs and one for installing the older z/XDC's SVCs. To facilitate running multiple XDCINIT PROCs, a suffix character can be added to the PROC name. Therefore, it is suggested that z1.13's XDCINIT job be named "XDCINITC".

Step 6: 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:

Go to z/XDC's Startup Panel, fill it in as shown in Figure 17 on page 39, 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 "*Renaming z/XDC*" on page 48), then the first thing you have to do is type z/XDC's new name (**Z1D**, 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 --------------------------------
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 XDC's Cross Domain Facility

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

    Tasklib DSNs  ===>
                  ===>
                  ===>
                  ===>
                  ===>
                  ===>
    or DDNAME     ===>

PARAMETER FOR OPTION 3 (BACKGROUND DEBUGGING VIA XDC-CDF):
  Connect to XDC-CDF using your current TSO Userid? (YES/NO) ===> YES
```

Figure 17 z/XDC's Startup Panel in ISPF

Go to ISPF's "Command Shell" (option =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 17** on page 39.

Go 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 (`DBCOLE.XDCZ1D.XDCLINK`, `DBCOLE.XDCZ1D.XDCLINKE` and `DBCOLE.XDCZ1D.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 18** on page 40 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. - 2003-2011. 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 Online Help, type HELP, then press ENTER. (Not pf-key 1)
.  DBC994I Type LIST HELP to display the Online Help Index.
.  DBC996I Type HELP HELP for how to use z/XDC's Online Help.

.  DBC830I XDC z1.13 ENTERED NONAUTHORIZED, AMODE(24), UNDER RB#3 FROM TCB#9 IN
.  DBC830I jobname (ASN=hhhh.nnn)
.  DBC891I XDC z1.13 ENTERED AS AN ESTAI OWNED BY TCB#8
.  DBC831I THE ERROR LEVEL AND RETRY LEVEL ENVIRONMENTS ARE THE SAME
_  DBC841I DEAD MSG: 7709 *** DBC901I SESSION IS READY FOR DEBUGGING, TYPE H AT
                                      THE LEFT FOR MORE INFORMATION ***


XDC ===> L PSW;L EPSW;L REGS
_    PSW 078D1000 00EBD000 (cc-LO 24) - IEFBR14+0
_   EPSW 078D1000 00060ABA (cc-LO 24) - DBC810W NOT WITHIN ANY IDENTIFIABLE
                                                MODULE OR ANY OTHER OBJECT
_       R0  00009B88 00061F90 E7C4C3C3 C1D3D340  *...h....XDCCALL *
_       R4  C9C5C6C2 D9F1F440 80FDA906 0005EE0D  *IEFBR14 ..z.....*
_       R8  00000000 00010220 0001BBF0 000606E0  *...........0...\*
_       R12 0005DE0E 00055E88 80060CEA 00E43000  *......;h.....U..*
```

Figure 18 Initial Screen Displayed by z/XDC with Fullscreen Display Support Installed

If, however, you see a display such as that shown in **Figure 19** on page 41, 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 XDCPANEL Panel or the XDCLBDEF Clist*" section on page 23 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 SPLIT and SWAP commands while using z/XDC.

```
TCB#9 RB#1 ------------------ z/XDC TPUT INTERFACE --------------------------
XDC ===>
   ISPP100  Panel 'XDCZ1DA' error
   ISPP100  PANEL NOT FOUND.

.  DBC750I PRESS ENTER TO CONTINUE DEBUGGING USING z/XDC'S FULLSCREEN TPUT
DISPLAY METHOD
```

Figure 19 Error When z/XDC is Not Properly Installed into ISPF

In any case, once you see the display in **Figure 18** on page 40, 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 0C1. 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 XDCCALL**A** and XDCCMD**A**, 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 20 on page 42, and press ENTER.** (In particular, 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 ISPF 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 (`DBCOLE.XDCZ1D.XDCLINK`, `DBCOLE.XDCZ1D.XDCLINKE` and `DBCOLE.XDCZ1D.XDCPLPA`) or load modules into the system's search order for your TSO session.

```
--------------------------- z/XDC STARTUP PANEL -------------------------------
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 XDC's Cross Domain Facility

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

  Tasklib DSNs  ===>
                ===>
                ===>
                ===>
                ===>
                ===>
  or DDNAME     ===>

PARAMETER FOR OPTION 3 (BACKGROUND DEBUGGING VIA XDC-CDF):
  Connect to XDC-CDF using your current TSO Userid? (YES/NO) ===> YES
```

Figure 20 Starting z/XDC Authorized from its Startup Panel in ISPF

Doing either of these things should cause the fullscreen display shown in **Figure 21** on page 43 to appear at your terminal. In particular, be sure that message DBC830I reports that z/XDC has been entered ***AUTHORIZED***. (If not, then you probably have not correctly updated the `IKJTSOxx` member of the `PARMLIB` libraries. Please see "*XDCCALLA and XDCCMDA: Additional Considerations*", on page 19, for more information.)

```
TCB#6 RB#1 ------------------ z/XDC TPUT INTERFACE ----------------------------
XDC ===>
.  DBC854I z/XDC for z/OS
.  DBC855I COPYRIGHT (C) COLESOFT PARTNERS, INC. - 2003-2011. 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 Online Help, type HELP, then press ENTER. (Not pf-key 1)
.  DBC994I Type LIST HELP to display the Online Help Index.
.  DBC996I Type HELP HELP for how to use z/XDC's Online Help.

.  DBC830I XXX z1.13 ENTERED *AUTHORIZED*, AMODE(24), UNDER RB#3 FROM TCB#6 IN
.  DBC830I DBCOLE3 (ASN=0042.3)
.  DBC891I XXX z1.13 ENTERED AS AN ESTAI OWNED BY TCB#5
.  DBC831I THE ERROR LEVEL AND RETRY LEVEL ENVIRONMENTS ARE THE SAME
_  DBC841I DEAD MSG: 7709 *** DBC901I SESSION IS READY FOR DEBUGGING, TYPE H AT
                                      THE LEFT FOR MORE INFORMATION ***


XDC ===> L PSW;L EPSW;L REGS
_    PSW 078D1000 00EBD000 (cc-LO 24) - IEFBR14+0
_   EPSW 078D1000 00066ABA (cc-LO 24) - DBC810W NOT WITHIN ANY IDENTIFIABLE
                                                MODULE OR ANY OTHER OBJECT
_       R0  00009B88 00061F90 E7C4C3C3 C1D3D340  *...h....XDCCALL *
_       R4  C9C5C6C2 D9F1F440 80FDA906 0005EE0D  *IEFBR14 ..z.....*
_       R8  00000000 00010220 0001BBF0 000606E0  *...........0...\*
_       R12 0005DE0E 00055E88 80060CEA 00E43000  *......;h.....U..*
```

Figure 21 Initial Screen Displayed by Authorized Mode z/XDC (with Fullscreen Support Installed)

Also notice that the title bar of the display reports "z/XDC **TPUT** INTERFACE" (not "z/XDC ISPF INTERFACE"). This is an ISPF limitation: ISPF simply does not permit authorized programs to use any of its services, including its Display Services. This means that it will not be possible to use ISPF's SPLIT and SWAP commands while debugging authorized programs running in TSO. (This restriction does not exist when using CDF to debug batch jobs. In that situation, it does not matter whether the target job is authorized or not. When you connect to such a debugging session via z/XDC's Startup panel [option 3], z/XDC will be able to use ISPF Display Services. See the next topic, "*Step 7: Installing z/XDC's Cross Domain Facility (CDF)*", on page 43, for more information.)

Again, you can terminate z/XDC via PF3 or PF4 or via an **END** command.

Step 7: Installing z/XDC's Cross Domain Facility (CDF)

z/XDC's "Cross Domain Facility" permits users at fullscreen terminals to debug abends that have occurred in background jobs and system tasks. CDF is implemented as a subserver managed by Server/XDC. The communication between a terminal user and a background job is accomplished via VTAM.

In order to use CDF, the computer center must create some VTAM definitions and also run a "life of IPL" task named xxxSRVER (where "xxx" matches z/XDC's name, see "*Renaming z/XDC*" on page 48). This contains the CDF subserver which connects interested and permitted users to jobs and tasks that have abended, passed control to z/XDC, and are waiting for someone to talk to.

IMPORTANT! The structure of CDF and the installation process for CDF changed in some major ways starting in z/XDC release z1.13. Customers upgrading from older releases need to start up z/XDC and read the topic **HELP WHATSNEW Z113 INCOMPATIBILITIES CDF** for detailed information concerning the changes and for how to continue to use CDF in a **compatibility mode**.

In any case, the current installation process for CDF requires the following steps:

1.) Member XSRVVTAM of `DBCOLE.XDCZ1D.XDCSRVER` contains a series of VTAM statements that define a major node named **XDCCDF** (**Figure 22** on page 44). This node is needed by the CDF subserver. These statements need to be copied to member xxxCDF of your system's VTAMLST library.

```
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 22 Typical VTAMLST File for XDC-CDF

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, then be sure to follow carefully the instructions given in the commentary contained within XSRVVTAM (but not shown in **Figure 22**).

3.) If you want the xxxCDF node to come up at VTAM startup time, then add xxxCDF's name (where "xxx" matches z/XDC's name) to the list of major node names found in the active `ATCCONxx` member of the VTAMLST library. Otherwise, the node will have to be started manually by an operator command ("`V NET,ID=xxxCDF,ACT`").

4.) Member XSRVPARM of `DBCOLE.XDCZ1D.XDCSRVER` contains control parameters (**Figure 23** on page 44) required by the CDF subserver. They tell CDF what the names are of the various APPLID's that are defined in XSRVVTAM.

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

Figure 23 Typical SYSIN File Parameters for XDC-CDF

These parameters need to be copied to member xxxSRVER of a parameter library (such as SYS1.PARMLIB). If you have changed the names of the APPLID's in XSRVVTAM, then you will have to make corresponding changes in XSRVPARM so that CDF will know what its APPLID names are. Follow carefully the instructions given in the commentary contained within XSRVPARM.

5.) A startup proc for the xxxSRVER started task, such as is shown in **Figure 24** on page 45, needs to be added to one of your system's PROCLIB's. The proc's name should be "xxxSRVER" (where "xxx" matches z/XDC's name). A model for the PROC can be found in the XSRVPROC member of DBCOLE.XDCZ1D.XDCSRVER. The JCL should be edited as indicated in **Figure 24**.

```
//xxxSRVER EXEC PGM=xxxSERVE,PARM='SERVER',REGION=0M
//STEPLIB  DD  DISP=SHR,DSN=<authorized libraries containing xxxSERVE>
//SYSIN    DD  DISP=SHR,FREE=CLOSE,DSN=<a parameter library containing parms for the CDF (and
                                        other) subservers>
//SYSPRINT DD  SYSOUT=*  (<--- optional DD card)
```

All occurrences of **xxx** should be replaced by z/XDC's name. (See "*Renaming z/XDC*" on page 48 for more information.)

The //STEPLIB statement will not be needed, of course, if xxxSERVE is installed into a linklist library.

Figure 24 Model JCL for Server/XDC

6.) Server/XDC needs to have READ access to the XDC.AUTH[31] or AUTH[32] rule.

7.) The CDF subserver task accepts logons from users in much the same way as does TSO or CICS. If your site has a security system, then the xxxSRVER task needs to be defined to that system such that security will permit the CDF subserver to make security calls to it in behalf of other users. Please refer back to "*Defining z/XDC to Your Computer's Security System*" on page 28 for details.

8.) In addition, the CDF subserver examines the batch jobs that are pending debugging and issues "RACHECK" type security calls to determine which batch jobs the logged on user is permitted to debug. This check is done before CDF displays to the user the list of jobs pending debugging[33]. 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 user. These violations can be disregarded since the user has no control over these access attempts.

The rule that CDF checks is "FASM.---" for CA-ACF2 and CA-Top Secret systems, and "XDC.FASM.---" for RACF systems. Refer back to "*Defining z/XDC to Your Computer's Security System*" on page 28 for more information.

[31] for RACF

[32] for CA-ACF2 or CA-Top Secret

[33] This is so that CDF will list only those jobs that the user is permitted to debug.

9.) The following operator commands need to be added to your system's IPL process:

- **`V NET,ID=xxxCDF,ACT`**
  If you have not added xxxCDF's name to the active `ATCCONxx` member of VTAMLST, then this command needs to be issued during or after VTAM's startup process. It activates the xxxCDF major node definition needed by the CDF subserver.

- **`S xxxSRVER`**
  This command starts Server/XDC. If xxxSRVER is started before the xxxCDF VTAM node comes up, then the CDF subserver will just wait for the node.

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 xxxSRVER,HELP" FOR A LIST OF VALID COMMANDS.
```

Step 8: CDF Installation Verification

In order to test CDF, you first must have a background job that has abended and is asking for CDF services. The job shown in **Figure 25** on page 46 will serve this purpose.

```
//      --- JOB ---
//IVP      EXEC PGM=xxxIVP
//STEPLIB   DD  DISP=SHR,DSN=DBCOLE.XDCZ1D.XDCLINK
//          DD  DISP=SHR,DSN=DBCOLE.XDCZ1D.XDCLINKE
//          DD  DISP=SHR,DSN=DBCOLE.XDCZ1D.XDCPLPA
//xxxPROF   DD  DISP=SHR,DSN=<your ISPF profile library>  (<-- optional DD card)
```

Where **xxx** matches z/XDC's name. (See "*Renaming z/XDC*" on page 48 for more information.)

Figure 25 Model JCL for Testing xxxCDF

The //xxxPROF DD card, if present, is used by z/XDC to find the user's profile data (member xxxXDC1D). 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 //xxxPROF is omitted, then z/XDC uses the "factory defaults" documented in **HELP PROFILES MENU**.

There are two ways users can log onto CDF:

- TSO users with ISPF can use z/XDC's Startup Panel ("XDCPANEL", see "*Adding XDCMLIB, XDCPLIB, and XDCTLIB*" on page 22). From that panel, select option **3** to connect to CDF.

- Other users can simply logon directly to CDF from an idle VTAM terminal. By default, the necessary command is "LOGON APPLID=xxx".[34, 35] CDF will then present the user with a screen requesting his userid and TSO password (**Figure 26** on page 47).

[34] At some Data Centers, the correct command would be "LOGON APPLID(xxx)".

[35] Where "xxx" matches the APPLID1 definition found in Server/XDC's //SYSIN parameters. If CDF has been installed as directed, then "xxx" will match z/XDC's name.

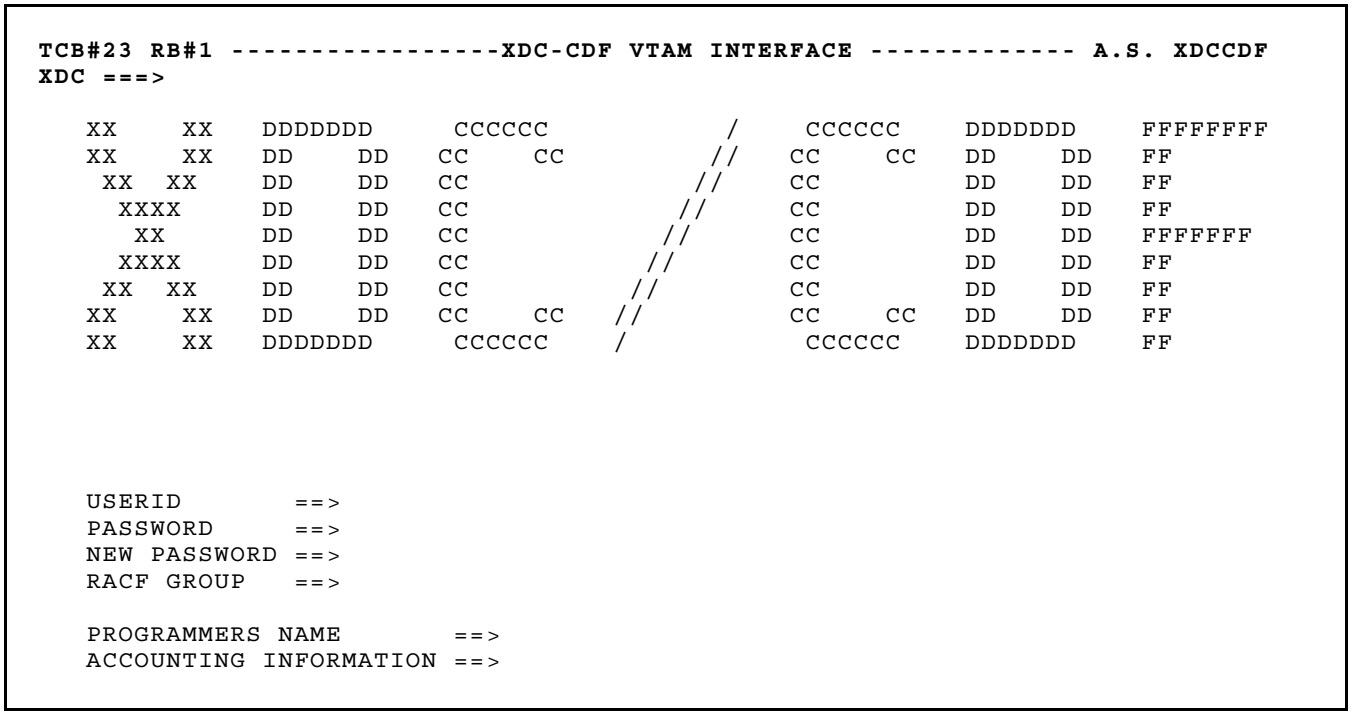

Figure 26 Logon Screen for VTAM connections to z/XDC's Cross Domain Facility

Once within CDF, 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 27** on page 47). 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. He will see a normal z/XDC debugging screen, and he will be able to issue normal z/XDC commands.

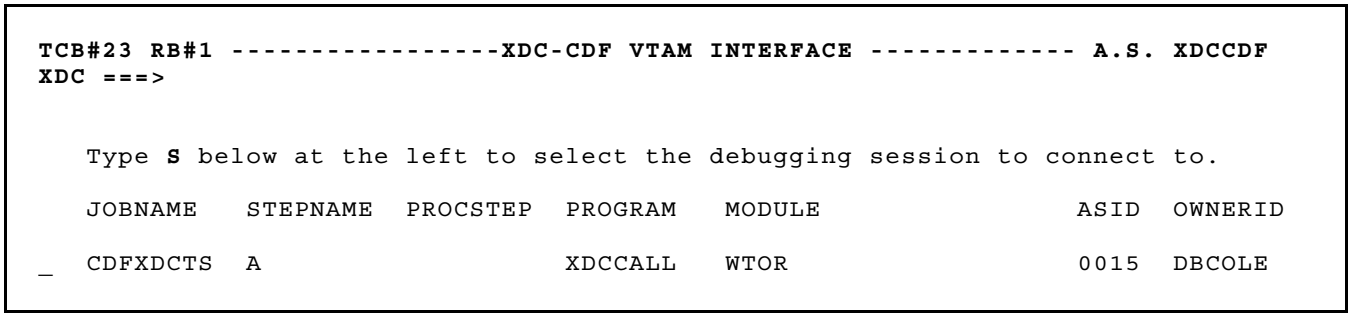

Figure 27 XDC-CDF Job Selection menu

For further information, though, the user should type **HELP CDF**[36] before proceeding.

[36] After selecting the job to be debugged, not before.

Step 9: Renaming z/XDC (Coexistence with Older XDCs)

If you want to install z/XDC z1.13 into your system and have it coexist with an older 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 XDC.) You may rename z/XDC to any other 3-character name you choose, but the recommended name would be **Z1D**.

The names of all z/XDC load modules in DBCOLE.XDCZ1D.XDCLINK, DBCOLE.XDCZ1D.XDCLINKE and DBCOLE.XDCZ1D.XDCPLPA start with the characters **XDC**. These first three characters can be changed to Z1D, to DBC, 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.
- The z/XDC load modules may be renamed directly within the DBCOLE.XDCZ1D.XDCLINK and DBCOLE.XDCZ1D.XDCPLPA libraries; but remember, in order to apply maintenance, you will have to rename them all back to **XDC**nnnnn, then apply the maintenance, and then rename them all back to their clone names. (At my own shop, I've written a clist, named $RENZ1D, to do this. That clist is shown in **Figure 28** on page 48. A copy of it can be found in DBCOLE.XDCZ1D.XDCCLIST.)

RENZ1D Clist

```
PROC 2 OLDNAME NEWNAME
CONTROL LIST

IF &SYSENV EQ FORE THEN +
  CALL 'SYS1.CSW.LINKLIB(PDS)' 'RENZ1D ''DBCOLE.XDCZ1D.XDCLINK'''
ELSE +
  PDS 'DBCOLE.XDCZ1D.XDCLINK'
REN &OLDNAME &NEWNAME GROUP

PDS 'DBCOLE.XDCZ1D.XDCPLPA'
REN &OLDNAME &NEWNAME GROUP
EN
```

This is a sample CLIST for renaming a z/XDC clone from one clone name to another. It uses an old shareware command named **PDS**[*]. Usage:

RENZ1D Z1D XDC renames all modules from Z1Dnnnnn to XDCnnnnn.
RENZ1D XDC Z1D reverses the rename.

If you don't have **PDS**, any similar command with equivalent capability will do.

———

[*] **PDS** can be found at the "CBT mods" web site: www.cbttape.org/freepds.htm.

Also, a vendor supported successor product named "**STAR TOOLS**" can be purchased from **Serena**: 800-457-3736. WEB: **www.serena.com**, then select "products", then "Star Tools".

Figure 28 RENZ1D Clist

- 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 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 use a clone name of **Z1D**.)

- User's will have to invoke z/XDC using clone names (**Z1D**nnnnn names) instead of XDCnnnnn names. Example:
```
//A EXEC PGM=Z1DCALLA,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'Z1D'
```

  Suggestion: Code your program to obtain z/XDC's name via a parameter, command or some other mechanism. My personal favorite way to do this would be to scan the TIOT for a ddname of the form `XDCIS`**`xxx`**, 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: `//XDCIS`**`Z1D`**` DD DUMMY`. For more information, see **HELP XDCCLONES #XDCIS**.

- Any `//XDCxxxxx` DD cards or dynamic allocations you might have for your debugging sessions (profiles, traces, other controls) will have to be changed. Examples:
```
//Z1DQUICK DD DUMMY
//Z1DPROF  DD DISP=SHR,DSN=userid.ISP.ISPPROF
```

- At IPL time, you will need to run a **Z1D**INITc proc similar to the XDCINITc proc described in *Step 5: Installing z/XDC's Service and Hook SVCs (via the XDCINITc Job)* (page 9). Proceed as follows:

  - Make a copy of your XDCINITc proc, and rename it to **Z1D**INITc.

  - Edit **Z1D**INITc to change all occurrences of XDC to **Z1D**.

  - Add a `//STEPLIB` card if necessary, but note that the libraries must all be APF authorized.

  - Add a command to your IPL process to automatically start up **Z1D**INITc, but take care to see that **Z1D**INITc does not run until after XDCINITc has started! (Normally, XDC's hook SVC is #199. If you allow **Z1D**INITc to run first, then SVC #199 will become **Z1D**'s hook SVC, and XDC's hook SVC will become #197. That might cause problems for some of your XDC users.)

- In ISPF, to use the **Z1D** name, at the XDC Startup Panel, you simply have to type **Z1D** on the command line and then press ENTER. The panel will automatically adjust itself to invoked **Z1D** instead of XDC.

- For CDF, you need to do the following:

  - Make a copy of your XDCSRVER startup proc, and rename both it and everything within it from `XDCxxxxx` to `Z1Dxxxxx`.

  - Make a copy of the `SYSIN` file referenced by your **Z1D**SRVER startup proc, and perform

similar renames within it.

- Make a copy of your XDCCDF VTAMLST member, rename it to **Z1D**CDF and perform similar renames within it.
- For auto-start of **Z1D**CDF's nodes at IPL time, add **Z1D**CDF to your VTAMLST(ATCCONxx) file.
- Also, add a S **Z1D**SRVER command to your PARMLIB(COMMNDxx) file.

- For TSO, you will have to add **Z1D**CALLA and **Z1D**CMDA to the AUTHCMD section of your active IKJTSOxx member of PARMLIB.
- Note, security system rules do **not** have to be specifically created or changed for any clone. All clones of XDC will honor security rules based upon the name **XDC** (not the clone's name).
- However, For sites that use the CA-ACF2 security system, the following actions have to be taken:
  - CA-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 **xxx**CALL, **xxx**CMD, **xxx**CALLA, **xxx**CMDA, **xxx**TFS, and **xxx** (where **xxx** is the desired clone name [**Z1D** in the current example]).
  - An **xxx**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.
- XDC PROFILES: When a user first starts up **Z1D**, if he had a saved profile in the older XDC, then **Z1D** will not be able to find it automatically. So to access his old profile, he will have to issue the following commands:

```
PROFILE READ XDC1C XDC
PROFILE SAVE
```

  The PROFILE SAVE command will cause the profile to be rewritten using the name **Z1D**XDC1D (z/XDC's default profile name when running under the clone name of **Z1D**).

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

Step 10: Making DBCOLE.XDCZ1D.XDCADATA the Default MAPLIB Library

The DBCOLE.XDCZ1D.XDCADATA library contains ADATA[37] for a few of z/XDC's load modules, the most important of which is the XDCMAPS load module. XDCMAPS contains mapping information about a very large number of IBM system control blocks. the DBCOLE.XDCZ1D.XDCADATA library contains source image information about those control blocks. Consequently, it is useful to set up DBCOLE.XDCZ1D.XDCADATA as a default place for z/XDC to look when it needs to build dsect maps in response to a DMAP command.

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 MAPLIB list that will be automatically available to a user's debugging session without requiring him to issue the SET MAPLIBS command. This

[37] **ADATA** is a file of data produced during the assembly or a program and that 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.

default MAPLIB information is kept in the user's session profile. The DBCOLE.XDCZ1D.XDCADATA (or whatever you have renamed it to) library is a good candidate for being a default MAPLIB library. To make it a default MAPLIB library, you will have to start a z/XDC debugging session and then issue the following commands:

```
SET MAPLIBS RESET DBCOLE.XDCZ1D.XDCADATA[38] SAVE
PROFILE SAVE
```

But don't actually issue these commands yet. Read the next section ("Changing the Default Profile: TFSXDC1D") first.

Step 11: Changing the Default Profile: TFSXDC1D

TFSXDC1D, as mentioned earlier, is a system-wide default profile for z/XDC (see "*Adding XDCMLIB, XDCPLIB, and XDCTLIB*" on page 22). When a user invokes z/XDC **for the first time**, TFSXDC1D is the profile that is loaded for his use.[39] 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 command, then z/XDC stores the changed profile into the user's own profile library in a member named xxxXDC1D, where "xxx" matches z/XDC's name (see "*Renaming z/XDC*" on page 48).

How to Change z/XDC's System-Wide Default Profile

If you wish to change the distributed default profile, then do the following:

- Review the **HELP FULLSCREEN PROFILE** topic in **z/XDC z1.13 User's Guide** (or in z/XDC's Online Help) to learn about the kinds of things that z/XDC saves in session profiles.
- Start up z/XDC. Your current profile will be loaded. (It doesn't matter whether you start z/XDC from within ISPF or from TSO's "READY" mode).
- Type **PROFILE READ XDC TFS**. This will force z/XDC to read the system default profile named TFSXDC1D from your ISPTLIB.
- Alternatively, type **PROFILE RESET** to cause z/XDC to reset all profile data to "factory defaults" values that are assembled directly into z/XDC itself (thereby bypassing TFSXDC1D).
- Alternatively, if you are using a TSO terminal having a large and wide display geometry, type **PROFILE READ WIDE XDC**. This will cause z/XDC to read a profile named TFSWIDE from your ISPTLIB. (For more information about terminals with large display geometries, start up z/XDC and then type **HELP FULLSCREEN TERMINALS GEOMETRIES**.)
- Issue the following command to make the DBCOLE.XDCZ1D.XDCADATA library the default MAPLIB[40]

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

[39] If a user has previously saved his own z/XDC profiles, then z/XDC will not automatically load TFSXDC1D Instead, it will load the user's profile and then automatically convert it, if necessary, to z1.13 format. But z/XDC will not write the converted profile back to disk unless and until the user issues a PROFILE SAVE command.

[40] A **MAPLIB library** is a library that contains ADATA files created by an assembler and from which z/XDC's MAP and DMAP commands can build Source Image Maps for use in Source Level Debugging.

for future debugging sessions:

```
SET MAPLIBS RESET DBCOLE.XDCZ1D.XDCADATA[41] SAVE
```

See the preceding section ("*Making* DBCOLE.XDCZ1D.XDCADATA *the Default MAPLIB Library*", page 50) for more information.

- Use z/XDC's **PROFILE** command (without operands) to enter into a menu system (see **Figure 29** on page 52) that allows you to display and change all[42] of z/XDC's current profile settings. Make any desired changes by simply overtyping the displayed profile values. For more information, see **HELP PROFILES MENU**.

```
TCB#6 RB#1 ------------------ z/XDC TPUT INTERFACE ----------------------------
XDC ===>
   Press PF3 or PF5 to accept changes and exit. (PF5 exits this dialog
   entirely.)

   The currently active profile is named XDC.
   It was read from member TFSXDC1D of DDname ISPTLIB.
   Profile data is saved only upon explicit request.
   Some profiled values have changed since this profile was read.
   Should this profile be saved now? ==>  NO
   Profile Description ==>  User provided descriptive comment

   Select one or more of the following menu options:

    _  Display/Change TSO/ISPF/CDF Related Settings.
    _  Display/Change Terminal Definitions.
    s  Display/Change Debugging Session PF-keys.
    _  Display/Change Online Help PF-keys.
    s  Display/Change Session Logging Parameters.
    _  Display/Change Window Control Parameters.
    _  Display/Change READ Command and ZAP Command Parameters.
    _  Display/Change FORMAT/TRACE/ZAP/FIND Parameters.
    s  Display/Change PRINT and Misc. Other Parameters.
```

Figure 29 z/XDC's Profile Menu System

- Exit the profile menus via **END** commands (PF-key 3) or a **QUIT** command.
- Type **PROFILE SAVE XDC TFS** to save the changed profile back into your profile library under the member named TFSXDC1D.
- Leave z/XDC and invoke ISPF's "Move/Copy Utility" (=3.3).
- Move (not just copy) your changed profile (member name TFSXDC1D) from your profile library to your system's ISPF table library.

Which System-Wide Default Profile Values You Might Consider Changing

As discussed above, it is important to set up the DBCOLE.XDCZ1D.XDCADATA library as a default MAPLIB library for use by your user community. If you haven't done so already, then please see "*Making*

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

[42] All except MAPLIB lists, that is.

DBCOLE.XDCZ1D.XDCADATA the Default MAPLIB Library" on page 50 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:

- **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 should change the default to the appropriate class. (See **Figure 30** on page 53.)

```
TCB#6 RB#1 ------------------ z/XDC TPUT INTERFACE ----------------------------
XDC ===>
   Press PF3 or PF5 to accept changes and exit. (PF5 exits this dialog
   entirely.)

   Currently, the log is open.


   SESSION LOGGING OPTIONS:
      Automatic session logging in effect? ==>  YES
      Number of scrollable messages        ==>  10000  (100-32767)
      Write log files to disk or spool?    ==>  SPOOL  (DISK, SPOOL)

   Output characteristics when logging to spool:
      Dataset SYSOUT class    ==>  X
      Place in Held status?   ==>  YES
      Printer destination     ==>
      Output Limit (OUTLIM)? ==>  0          (0 or 1000-16777215)

   Output characteristics when logging to disk:
      Unparsed DSNAME         ==>  *P.*XLOG.*D.*T
      Resolved DSNAME         ==>  DBCOLE.XDCLOG.D110111.T091511
      Allocation volume       ==>
      Allocation unitname     ==>
      Append or overwrite?    ==>  APPEND       (APPEND, OVERWRITE)
```

Figure 30 Profile Settings for z/XDC's Session Log

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

- **xxxPRINT File Characteristics:** z/XDC supports a **PRINT** command that users can use to print various things such as selected portions of the Online 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 31** on page 54.) You may want to check out these settings and perhaps change them if necessary to match the characteristics of your most typical printers.

```
TCB#6 RB#1 ------------------ z/XDC TPUT INTERFACE ---------------------------
XDC ===>
   Press PF3 or PF5 to accept changes and exit. (PF5 exits this dialog
   entirely.)

   PRINT AND MISC. OTHER OPTIONS                    CHOICES
    60     XDCPRINT lines per page:                 10-255 or 0 (no pagination)
    X      XDCPRINT and SYSUDUMP output class:      A-Z, 0-9, or *
    DEFAUL XDCPRINT output status:                  HOLD   NOHOLD   DEFAULT
    YES    Upcase csect names and quoted strings?   YES    NO
    NO     Intercept/debug CANCEL/DETACH abends?    YES    NO
    64BIT  Meaning of "!" indirect operator:        AMODE  64BIT
    5      Multitasking timeout interval:           1 to 3600 seconds.
    ~      Builtin Function Leader Character:       ¢ ` ~
```

Figure 31 Profile Settings for xxxPRINT Output Files

- **Individual PF-key Assignments:** Many of z/XDC's factory default PF-key assignments are "real nonstandard". (See **Figure 32** on page 55.) 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 SCREEN 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 SCREEN CREATE** the default setting for PF1, **not** HELP.

```
TCB#6 RB#1 ------------------ z/XDC TPUT INTERFACE ----------------------------
XDC ===>
   Press PF3 or PF5 to accept changes and exit. (PF5 exits this dialog
   entirely.)

    _  Select here to view and update the PF-key sets to ranges assignments.

   PFK#  SET-A
    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);GOT
   12    12TH RETRIEVE


                                      - - -


TCB#6 RB#1 ------------------ z/XDC TPUT INTERFACE ----------------------------
XDC ===>
   Press PF3 or PF5 to accept changes and exit. (PF5 exits this dialog
   entirely.)

    _  Select here to view and update the PF-key sets to ranges assignments.

   PFK#  SET-B
   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
```

Figure 32 Default PF-key Definitions

As I said, some of the defaults are real nonstandard, but there are good reasons for this.

Before making any changes to the installation defaults, please read the **HELP FULLSCREEN PFKEYS DFLTKEYS** topic (in **z/XDC z1.13 User's Guide** or in z/XDC's Online Help) for the detailed descriptions of what the default PF-keys are and do.

Refer to the **HELP FULLSCREEN PROFILE** topic in **z/XDC z1.13 User's Guide** or in z/XDC's Online Help for more detailed information about user profiles.

Step 12: 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 z1.13 Release Guide** for more information. Also, check out the commentary located in the #DBCPARM macro (found in the `DBCOLE.XDCZ1D.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. "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 z1.13 User's Guide** or in z/XDC's Online Help. The supported User Exits are:

- **A User Communications Interface ("UCI") Exit**
  This 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. 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 z1.13 User's Guide** or in z/XDC's Online Help.

  Source code for a sample UCI exit can be found in `DBCOLE.XDCZ1D.XDCASM(SRCONLHG)`.

- **Consolidated Exits**
  z/XDC currently supports the following five consolidated exits:

  - **An "Initialization" Exit**
    This 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**
    This 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**
    This exit is called every time z/XDC is about to return to its caller with "percolate" signaled.

  - **A "User SVC" Exit**
    This exit is called by z/XDC when it is processing a TRACE command and the current instruction is a user SVC (or an "EX" 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**
    This 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 `DBCOLE.XDCZ1D.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 z1.13 User's Guide** or in z/XDC's Online Help.

Source for a sample router module can be found in `DBCOLE.XDCZ1D.XDCASM(SRCXITSR)`.

- **A "Front-end/Back-end" Routine**
  Every time z/XDC receives control, it GETMAIN's and builds an 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 z1.13 User's Guide** or in z/XDC's Online Help for more information.

INSTALLING A DEBUGGING INTERFACE INTO JES2

DBCOLE.XDCZ1D.XDCJES2 contains an interface for using z/XDC to debug JES2 code. This interface will fit all releases of JES2 from MVS/370 SP 1.3.6 through at least the z/OS 1.3 release of JES2 (and probably beyond).

Please understand that this interface is necessary only if you intend to use z/XDC to debug JES2 code and JES2 exit routines. If you do not intend to write any mods or exits for JES2, then you need not install this interface, and you can skip reading this section.

If you decide to install this interface, then you should also make a copy of this section of this manual available to your JES2 programmers. It contains information that they will need to know. Also they should read the **HELP DEBUGGING JES2** topic in **z/XDC z1.13 User's Guide** or in z/XDC's Online Help.

What the XDC/JES2 Interface Is and Does

The XDC/JES2 interface allows the customer to use z/XDC to debug mods and exit routines running under JES2's main task.[43] This interface consists of three components: two JES2 exits and an optional source code update file. The two exits are an Exit 0 for implementing an "XDC" initialization option and an Exit 5 for implementing a "$XDC" command. The source code modification makes changes to JES2's "$CB" and "$MODULE" macros in support of z/XDC's csect mapping command.

Each of the three components of the XDC/JES2 interface is completely independent of the other two. A customer may freely install or not install any or all of the components in any combination.

The Members of DBCOLE.XDCZ1D.XDCJES2

The XDCJES2 library contains the following members for the JES2 interface:

J2IFXIT0 and **J2IFXIT5**
Contains source code for a JES2 Exit 0 and an Exit 5 routine, respectively.

J2IFASM0 and **J2IFASM5**
Contains sample JCL for assembling J2IFXIT0 and J2IFXIT5, respectively.

J2IFLKD0 and **J2IFLKD5**
Contains sample JCL for linkediting J2IFXIT0 and J2IFXIT5, respectively. Please note that these two exits must be linkedited as "serially reusable" (i.e. "PARM='REUS...'") and not as reentrant.

J2IFUPDT
Contains a source code update to JES2's $CB and the $MODULE macros.

J2IFUJCL
Contains sample JCL for applying J2IFUPDT to JES2's source library.

J2IFHASM
Contains sample JCL (a) for creating a HASPDOC load module containing a full set of JES2 control block maps for use by z/XDC, and (b) for reassembling any or all of JES2's source modules with

[43] Currently, the interface does not directly support debugging exits running under JES2 subtasks. To debug such exits, follow the procedures outlined in z/XDC's Online Help. Start z/XDC and then type HELP DEBUGGING EXITROUTINES.

csect maps for use by z/XDC and with redundant dsect maps edited out.

Exit 0: The "XDC" Initialization Option

Exit 0 adds support permitting a system operator to include "XDC" in his reply to JES2's initialization WTOR ("$HASP426 SPECIFY OPTIONS - JES2 SP v.r.m"). If the operator does specify "XDC", then an exit routine will locate and load into storage a copy of z/XDC, issue an ESTAE macro to make z/XDC the newest ESTAE routine, and then execute a #DIE macro to breakpoint to z/XDC. At this point a JES2 programmer can use z/XDC's commands to load maps, set breakpoints, etc. Then when he is ready, he can issue z/XDC's "GO" command to return control back to JES2 which will then proceed with its initialization.

This "XDC" initialization option can be used by the JES2 programmer to turn the z/XDC debugging interface on and to pass control to z/XDC as early as possible in JES2 initialization. From this point z/XDC commands can be used to set breakpoints for the debugging of JES2 initialization routines as well as any other JES2 main task routines.

Due to JES2 environment limitations existing at the time that JES2 searches for and invokes Exit 0 routines, the load module containing z/XDC's Exit 0 must be named exactly "HASPXIT0". If the customer already has another Exit 0 routine, then that routine must be merged with z/XDC's Exit 0. For more information, please refer to the appropriate **User Modifications and Macros** manual for your release of JES2. Also, refer to commentary located within the source code for z/XDC's Exit 0.

The source code for z/XDC's Exit 0 is contained in "J2IFXIT0". Sample JCL for assembling the exit is contained in "J2IFASM0". Sample JCL for linkediting the exit is contained within "J2IFLKD0". Please note that this exit must be linkedited as "serially reusable" (i.e. "PARM='REUS...'") and not as reentrant. Also, it is recommended that this exit be assembled and linkedited with PARM=TEST.

JES2 initialization parameter cards are not needed for defining EXIT 0. This is because JES2 searches for and invokes Exit 0 routines long before it processes its initialization parameter cards.

For additional technical information, please see commentary found within the source code located in J2IFXIT0.

Exit 5: The "$XDC" Command

Exit 5 adds support for a "$XDC" command and makes a change to the handling of the "$PJES2,ABEND" command.

The "$XDC" command can be used to turn on, turn off, and query the status of z/XDC debugging within JES2 as follows:

> "**$XDC ON**" or just "**$XDC**"
> This command turns on z/XDC debugging within JES2's main task. First, it checks to see if z/XDC has already been loaded into storage and established as the JES2 main task's newest ESTAE routine. If not, then it loads a copy of z/XDC into storage and then issues an "ESTAE" macro to make it the newest ESTAE routine.
>
> Next, the command attaches a subtask that traps to z/XDC via a #DIE macro (available from `DBCOLE.XDCZ1D.XDCMACS`) to pass control to z/XDC. At this point, z/XDC commands can be used to load maps, set breakpoints, etc. When a GO command is issued, the subtask terminates until the next $XDC ON command is issued.

"**$XDC OFF**"
This command turns z/XDC debugging off in JES2. If z/XDC currently is the newest ESTAE routine, then this command issues an "ESTAE 0" macro to cancel z/XDC and then a "DELETE" macro to remove z/XDC from the address space.

"**$XDC STATUS**" or "**$XDC ?**"
This command displays messages reporting the current status (on or off) of the z/XDC interface.

No ill effects arise from issuing multiple "$XDC ON"s and/or "$XDC OFF"s in a row. "$XDC ON" will always breakpoint to z/XDC, but it will not create excessive z/XDC ESTAE's. "$XDC OFF" will delete the newest ESTAE only if it is z/XDC.

When the z/XDC interface is on, any abends, $ERRORs, and/or breakpoints that occur within JES2's main task[44] will cause control to pass to z/XDC. On the other hand, when the interface is off, all abends, $ERRORs, and breakpoints will cause control to be passed directly to JES2's "Catastrophic Error" routine, "$ABEND" (which will see breakpoints as nothing more than 0C1 abends).

The source code for z/XDC's Exit 5 is contained in "J2IFXIT5". Sample JCL for assembling the exit is contained in "J2IFASM5". Sample JCL for linkediting the exit is contained within "J2IFLKD5". Please note that this exit must be linkedited as "serially reusable" (i.e. "PARM='REUS...'") and not as reentrant. Also, it is recommended that this exit be assembled and linkedited with PARM=TEST.

To activate z/XDC's Exit 5, the following parameter cards need to be added to JES2's "HASPPARM" file:

```
LOADMOD(XDCEXIT5)
EXIT5 ROUTINE(EXIT5XDC)
```

For additional technical information, please see commentary found within the source code located in J2IFXIT5.

Exit 5: The "$PJES2,ABEND" Command

When z/XDC's interface is on, all abends, including $ERROR abends, occurring within JES2's main task are intercepted and handled by z/XDC instead of by JES2's own routines. This includes the "PJ2" $ERROR that occurs when the "$PJES2,ABEND" command is issued.

When z/XDC intercepts and handles an abend (such as the one created by the "$PJES2,ABEND" command), it is possible that the JES2 programmer might issue z/XDC's "GO" command to cause JES2 to resume execution as if the abend had never happened. JES2's own implementation of the "$PJES2,ABEND" command, however, does not anticipate this possibility, and so it has no way to return control to JES2's dispatcher to resume normal processing.

The implementation of the "$PJES2,ABEND" command within z/XDC's Exit 5 provides the following way to resume normal JES2 execution, should the JES2 programmer so desire:

- When z/XDC gains control as a result of a $PJES2,ABEND command, issue a "MAP XDCEXIT5+0" command to load both a linkedit map and a csect map for z/XDC's Exit 5 routine.
- Issue a "S Q XDCEXIT5" to make Exit 5 the default load module and csect.

[44] Currently, the interface does not directly support debugging exits running under JES2 subtasks. To debug such exits, follow the procedures outlined in z/XDC's Online Help. Start z/XDC and then type HELP DEBUGGING EXITS.

- Issue "GO .PJ2RESUM" to pass control from z/XDC back to JES2.

The "$PJES2,ABEND" command is the only form of the $PJES2 command that is handled by z/XDC's Exit 5. All other forms of the $PJES2 command are handled by JES2's own routines.

Debugging JES2 with z/XDC Clones

By default the z/XDC interface implemented via the above described Exit 0 and Exit 5 will load and use a load module named "XDC". However, z/XDC contains extensive support permitting it to be renamed to any other 3-character name. When this is done, the renamed z/XDC is referred to as a "clone" of z/XDC. See "*Renaming z/XDC*" on page 48 for more information.

If you need to have the XDC/JES2 interface look for and use a z/XDC clone, then you must add the following DD card to JES2's startup JCL:

```
//XDCISxxx DD DUMMY
```

where "xxx" is replaced by the z/XDC clone's name. For example, if z/XDC has been renamed at your shop to "Z1D", then you will need to add:

```
//XDCISZ1D DD DUMMY
```

to JES2's startup JCL.

The presence of a "`//XDCISxxx`" DD card in JES2's startup JCL will have the following effects of z/XDC's interface:

- Both Exit 0 and Exit 5 will look for and load "xxx" instead of "XDC".
- Exit 0 will change the name of the z/XDC initialization option from "XDC" to "xxx".
- Exit 5 will change the name of the $XDC command from "$XDC" to "$xxx".

It is the JES2 programmer's responsibility to choose a clone name that does not interfere with other JES2 initialization options and operator commands.

Communicating with the XDC/JES2 Interface

When running in a **non-TSO** address space, z/XDC has available to it two possible mechanisms for communicating with a debugging programmer:

- Linemode style WTO/WTOR's to a selected operator's console,
- Or fullscreen TPUT/TGET's via z/XDC's "Cross Domain Facility" ("CDF") to a VTAM or TSO terminal. z/XDC can be setup to use either mechanism in JES2 regardless of whether the JES2 being debugged is a primary or a secondary JES2.

The linemode style WTO/WTOR interface is a fallback method of communication. The preferable method is the fullscreen interface via the Cross Domain Facility. In order to use CDF, however, all of the following conditions must be met:

- z/XDC's CDF component must be licensed and installed to run at your site.
- Server/XDC must be running. (Note that one implication of this requirement is that it is not possible

to use fullscreen communications when debugging JES2 initialization routines in the primary JES2, since at that point in time it will not yet have been possible to start Server/XDC. It is, however, possible or use z/XDC to:

- Debug anything else in the Primary JES2,
- Debug the initialization routines [as well as all the rest of] a secondary JES2.)

- A "`//xxxWTOR DD DUMMY`" card must **NOT** be included in JES2's startup JCL. (This would tell z/XDC to communicate via WTO/WTORs instead of via CDF.)

- An "`//xxxPROF`" DD card also should be included that points to a card-image library that contains a copy of the JES2 programmer's "xxxXDC1D" member from his own ISPF profile library. This contains the PF-key, screen layout, and other definitions needed by xxxCDF for the fullscreen support.

For more information concerning the Cross Domain Facility, please refer to z/XDC's Online Help. (Start z/XDC and then type "HELP CDF".)

J2IFUPDT: JES2 Csect Mapping Support

J2IFUPDT contains an IEBUPDTE format update file that contains certain source code modifications to JES2's $CB and $MODULE macros. This source code update is optional. It is provided to facilitate the process of making JES2 csect maps available for use by z/XDC during JES2 browsing or debugging. This update should be applied only if you are making source code mods to JES2. If you are just writing JES2 exit routines, then there is no reason to apply this update.

This update file correctly fit all versions and releases of JES2 from MVS/370 SP 1.3.6 through at least MVS/ESA SP 4.3.0 (and probably beyond).

These updates to the $CB and $MODULE macros cause the JES2 control blocks portion of all assemblies (except the HASPDOC assembly) to be bracketed with "SYMDEL" and "SYMNODEL" dsect cards. The presence of the SYMDEL and SYMNODEL dsect cards has no effect whatsoever on JES2 object code. This update does not in any way affect JES2's logic or behavior. Therefore, no production facility can ever become dependent on this update.

However, if the JES2 code is assembled with "PARM='TEST ...'", then the names "SYMDEL" and "SYMNODEL" will appear in the object deck symbol table (card type "SYM") that is generated by the Assembler. If, prior to linkedit, the object deck is then run through a post processing utility named "XDCSYMED", then all symbols occurring between the SYMDEL and SYMNODEL names will be purged from the object deck. This reduces the size of each object deck by hundreds of cards, and eventually when you linkedit the object decks together (using "PARM='TEST ...'" on the linkedit) to build the various JES2 load modules, their on-disk sizes will be reduced by literally thousands of card images (as compared to not using XDCSYMED and the SYMDEL/SYMNODEL dsect cards). See z/XDC's Online Help for more information about using "XDCSYMED". (Start z/XDC and then type "HELP MAPS XDCSYMED".)

Note, this process for eliminating excess symbol table entries is necessary because the Binder itself does nothing whatsoever to recognize and eliminate redundant "SYM" cards and entries.

The size and logic of load modules read into storage are never affected by the presence or absence of "SYM" records. This is because the system's program fetch routine ("IEWFETCH") always ignores SYM records.

The updates to $CB and $MODULE are contained in "J2IFUPDT". Sample JCL for applying the update is contained in "J2IFUJCL". Sample JCL for assembling and then symbol editing JES2 modules is contained

in J2IFHASM.

Using JES2 Csect Maps

You may or may not decided that you want to have csect maps available to z/XDC during JES2 debugging. If you are just writing JES2 exit routines, then you probably will not need JES2 csect maps. On the other hand, if you are writing mods to JES2 source code, then csect maps will be very useful to you.

If you do wish to use csect maps, then you might decide to generate them for either all or only a few JES2 source modules. To generate csect maps, you must do the following:

- First, you must apply the update found in J2IFUPDT to JES2's source library, SYS1.HASPSRC.
- Then you must assemble the desired one, several, or all JES2 source modules with PARM=TEST specified. Then you will have to linkedit the affected JES2 load modules also with PARM=TEST specified.
- You must assemble and linkedit the desired JES2 modules outside of SMP/E. This is because SMP/E does not support PARM=TEST.
- In order to reduce the on-disk size of the JES2 object and load libraries by many many cylinders, you will have to append to your assembly JCL a second step to use the XDCSYMED utility to remove excess SYM table entries from the object decks. This is because most or all of the JES2 source modules assemble approximately the same very large set of control block macros, and this leads to huge amounts of redundant information in the object modules. Worse, when the Binder combines the object modules into load modules, it takes no steps whatsoever to identify and remove redundant SYM table entries. It merely propagates them wholesale into the load module.
- In order to make XDCSYMED operate effectively, pairs of "SYMDEL/SYMNODEL" DSECT cards have to be added to the JES2 source modules to delimit for XDCSYMED those SYM table entries that are to be removed from the object decks. This is what the J2IFUPDT update does to the $CB and $MODULE macros.

J2IFHASM Contains sample JCL for reassembling all of JES2's source modules with PARM=TEST specified and with XDCSYMED invoked to edit the object modules. This leads to the creation of JES2 object and load modules with csect maps available for use by z/XDC and with redundant dsect maps edited out.

Using JES2 Dsect Maps

The generation of dsect maps for JES2 control blocks is a separate and somewhat simpler process than generating csect maps. All you have to do is assemble and linkedit HASPDOC with PARM=TEST specified. This creates a load module that contains a full set of JES2 control block maps in a form usable by z/XDC. Doing this assembly and linkedit will in no way affect JES2 processing. In fact JES2 will never load, make use of, or even know of the existence of the HASPDOC load module. However, when you use z/XDC to browse or debug JES2, you will find the control block maps provided by HASPDOC to be immensely useful in identifying and referencing the many control structures found within the JES2 address space.

J2IFHASM contains sample JCL for creating a HASPDOC load module containing a full set of JES2 control block maps for use by z/XDC. Note that when creating the HASPDOC load module, it is not

necessary (although it doesn't hurt) to apply the J2IFUPDT updates against the $CB and $MODULE macros. It also is not necessary (but again, it doesn't hurt) to process the HASPDOC object deck through the XDCSYMED utility.

Browsing JES2 via Foreign Address Space Mode

The XDC/JES2 interface described above is "invasive"; i.e., when used it can interrupt JES2 main task processing: printers stop, remotes time out, internal and other readers lock up, TSO logons also lock up, etc. This interface can be extremely useful during program development, and it can certainly be used freely against a secondary, non-production JES2 or on a private "sandbox" system. It is not, however, appropriate for use on a production system.

If all you want to do is to browse JES2's address space without affecting its processing, z/XDC's "Foreign Address Space Mode" is an excellent tool to use. All you need to do is:

- Logon to your TSO terminal,
- Invoke an authorized z/XDC debugging session ("XDCCALLA IEFBR14"),
- Delete z/XDC's lower display screen (move the cursor down and press PF13),
- Issue "SET ASID JES2".

At this point you are connected to JES2's address space, and you can use z/XDC's DISPLAY, FORMAT, LIST, EQUATE, MAP, DMAP, USING, and ZAP commands to display, map, format, and even sometimes repair all structures in JES2's private storage.

A Browsing Example

As an example, try the following command sequence:

`XDCCALLA IEFBR14`	Starts up an authorized debugging session.
`SET ASID JES2`	Switches to Foreign Address Space Mode against the JES2 address space.
`LIST TASKS`	Displays JES2's current subtask structure. Also (re)creates "TCB#n" equates.
`LIST RBS TCB#3`	Displays the RB's (usually just one) currently queued to JES2's main task. Also (re)creates "RB#n" equates.
`LIST REGS RB#1`	Displays the current registers being used (at least for the then current few nanoseconds) by the oldest RB running under JES2's main task.
`LIST PSW RB#1`	Ditto for the PSW.
`D HASJES20.X#1`	Displays JES2's HCT, unformatted.
`DMAP HASPDOC.HCT`	Loads a map of the HCT.
`USING HCT +0`	Bases the HCT map on top of the actual HCT.
`F +0,500`	Formats and displays lots of the HCT.

`EQ JOBQ .$JOBQPTR?`	Marks the start of JES2's job queue.
`DMAP .JQE`	Loads a JQE map.
`FIND 'MYJOBID' JOBQ`	Locates a JQE of interest.
`USING .JQEJNAME +0`	Bases the JQE map, properly aligned, onto the actual JQE.
`F JQE 50`	Formats a portion of the JQE map.

.
.
.

etc.