z/XDC®
INSTALLATION GUIDE

z/XDC® Release z2.2 for z/OS

David B. Cole

Generated December 3, 2022 (4:02am)

PREFACE

PROPRIETARY LEGEND

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

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 The Product may not be published, reproduced, distributed or made available to third parties for any purpose without the expressed written permission of Owner or Licensor.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

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

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

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

CONTACTING COLESOFT

The **z/XDC®** products are marketed by **ColeSoft Marketing, Inc** with its principal office in Charlottesville, 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:	**www.colesoft.com**

Our Technical Support contacts are:

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

Our Customer Services contacts are:

E-Mail: **support@colesoft.com**
Home Page: **www.colesoft.com**

Our snail mail address is:

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

ONLINE PRESENCE

ColeSoft Marketing maintains the following resources on the Internet:

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

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

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

TRADEMARKS

TFS™, **XDC-TFS™**, **server/XDC™**, **CDF™**, **XDC-CDF™**, **cdf/XDC™**, **FASM™**, **base/XDC™**, **c/XDC™** and **asm/XDC™** are trademarks of ColeSoft Partners, Inc.

Extended Debugging Controller®, **XDC®** and **z/XDC®** are registered trademarks of ColeSoft Partners, Inc.

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

PRODUCT MANUALS

z/XDC's documentation consists of the manuals shown below. The manuals are provided in PDF format.
All manuals can be downloaded from colesoft.com > Training > Documentation.

z/XDC z2.2 Manual Name	Description
Install Guide (the current document)	Describes how to install z/XDC
User Guide	Explains how to use z/XDC
Commands Reference	Describes all of z/XDC's commands
Messages Reference	Explains all numbered messages issued by z/XDC
Maintenance and New Functionality Guide	Describes what is new if z/XDC (updated constantly)
Primer	A basic tutorial to help the beginner get started with z/XDC
Quick Reference	A comprehensive syntax reference for all of the z/XDC commands

Figure 1 Available z/XDC Manuals

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 the above manuals from our web site (www.colesoft.com). Each manual is available in `PDF` format.

z/XDC® z2.2 INSTALLATION GUIDE

(Preface)

CONTENTS

FIGURES

z/XDC® z2.2 INSTALLATION GUIDE

THE Z/XDC PRODUCT SHIPMENT PACKAGE

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

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

The Shipment Package can be used:

- By new customers wishing to perform an initial product install of z/XDC at the latest maintenance level,
- By existing customers wishing to perform a complete product reinstall of z/XDC at the latest maintenance level,
- By existing customers wishing to update an existing product, bringing it up to the latest maintenance level.

Complete information about downloading and using the Shipment Package can be found in a Read Me file located at colesoft.com > Support > Shipment Package. Please download and review that document for all the step-by-step details.

Once you have completed that process, you are ready to begin the Integration Phase of the z/XDC installation. That documentation follows below.

Important! If you are applying a maintenance update, when you are done, there are certain reintegration steps that must **always** be done:

- You must always reintegrate all load modules. For details, see:
  - **Adding XDCPLPA Load Modules to Common Storage** on page 2,
  - and **Adding the XDCLINK and XDCLINKE Libraries to Your Linklist** on page 3.
- If the maintenance update has made any significant other changes, those will have to be reintegrated as well.
- Once you have completed all required reintegrations, **always** bounce server/XDC (see **Installing z/XDC's Server Job** on page 22).
- This is necessary to cause z/XDC's System Interface to be reinstalled. (See **Installing z/XDC's Service SVC, Legacy Hook SVC and System Interface** on page 21.)

Z/XDC SYSTEM INTEGRATION

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

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

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

There are the three load libraries:

- `DBCOLE.XDCZ22.XDCLINK`
- `DBCOLE.XDCZ22.XDCLINKE`
- `DBCOLE.XDCZ22.XDCPLPA`

And six FB-80 libraries:

- `DBCOLE.XDCZ22.XDCCLIST`
- `DBCOLE.XDCZ22.XDCCMDS`
- `DBCOLE.XDCZ22.XDCMLIB`
- `DBCOLE.XDCZ22.XDCPLIB`
- `DBCOLE.XDCZ22.XDCSRVER`
- `DBCOLE.XDCZ22.XDCTLIB`

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

- `DBCOLE.XDCZ22.XDCADATA`
- `DBCOLE.XDCZ22.XDCASM`
- `DBCOLE.XDCZ22.XDCDWARF`
- `DBCOLE.XDCZ22.XDCJCL`
- `DBCOLE.XDCZ22.XDCMACS`
- `DBCOLE.XDCZ22.XDCMMEF`
- `DBCOLE.XDCZ22.XDCSAMP`

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

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

Adding XDCPLPA Load Modules to Common Storage

`DBCOLE.XDCZ22.XDCPLPA` contains load modules XDC and XDC31.

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

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

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

- By copying the modules to an existing PLPA library, and then reIPLing.
- By adding DBCOLE.XDCZ22.XDCPLPA's dsname to the PLPA library list, and then reIPLing.
- By adding a COM='SETPROG LPA,ADD,MODNAME=(XDC,XDC31),DSN=DBCOLE.XDCZ22.XDCPLPA' command to an active COMMND*xx* member of your PROCLIB libraries to dynamically load the XDC and XDC31 load modules into the system's DLPA at IPL time. *[This is the recommended method.]*
- Note, COMMND*xx* files are processed only at IPL time, but a reIPL is not necessary since you can also issue the SETPROG LPA command manually at any time.)

The XDC and XDC31 load modules will have to be reinstalled into the PLPA or DLPA **every time** you apply z/XDC maintenance to the DBCOLE.XDCZ22.XDCPLPA library! That's because every maintenance update will make at least systemic changes to all load modules.

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

```
SETPROG LPA,ADD,MOD=(XDC,XDC31),DSN=DBCOLE.XDCZ22.XDCPLPA
DISPLAY PROG,LPA,MOD=XDC
DISPLAY PROG,LPA,MOD=XDC31
```

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

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

- The newer instances will supercede the older ones in the system's DLPA search order.
- But if you wish to recover wasted CSA storage, then after issuing the SETPROG LPA,ADD command to load the newer instances into the DLPA, you can then issue a SETPROG LPA,DELETE,*operands* command to delete the older instances of XDC and XDC31. The precise command is:
    - SETPROG LPA,DELETE,MOD=(XDC,XDC31),FORCE=YES,OLDEST

- Issue the above command repeatedly until it fails (in case multiple older instances exist)[4].

Adding the XDCLINK and XDCLINKE Libraries to Your Linklist

Both DBCOLE.XDCZ22.XDCLINK and DBCOLE.XDCZ22.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 libraries.
- (But see footnote[5] for one exception).
- XDCLINKE is a PDSE that contains program objects that may not reside in PDSs.

---

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

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

[4] Use of the OLDEST operand keeps the command from removing the newest instance, i.e. the instance that you just loaded with the SETPROG LPA,ADD command.

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

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

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

Copying XDC Load Modules into Existing Linklist Libraries

If you elect to copy the `DBCOLE.XDCZ22.XDCLINK` and `DBCOLE.XDCZ22.XDCLINKE` load modules into existing linklist libraries:

- After copying the z/XDC load modules, perform an `LLA REFRESH` to make them usable (assuming the target linklist libraries did not create and expand into extra extents).

- The copying process may cause your target linklist libraries to create and expand into extra extents. If so, that will cause programs that try to use those load modules to fail with IO errors. But don't worry. It's recoverable.

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

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

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

- This copying process will have to be repeated every time you apply z/XDC maintenance.

Adding XDC Load Libraries to the Linklist Concatenation

If you elect to add the `DBCOLE.XDCZ22.XDCLINK` and `DBCOLE.XDCZ22.XDCLINKE` libraries to the linklist concatenation, then:

- Add `DBCOLE.XDCZ22.XDCLINK`'s and `DBCOLE.XDCZ22.XDCLINKE`'s dsnames to the LNKLST section of an active PROG*xx* member of the PARMLIB libraries.

- If your active IEASYS*xx* member of PARMLIB has `LNKAUTH=APFTAB` specified, then the `DBCOLE.XDCZ22.XDCLINK` and *ditto*`.XDCLINKE` libraries will have to be made APF authorized by adding their names to the APF section of an active PROG*xx* member of the PARMLIB libraries.

- Historically, when you change the linklist, you need to wait for an IPL before those changes are activated. However, these days you can use the `SETPROG APF,`*operands* and `SETPROG LNKLST,`*operands* commands to authorize the `DBCOLE.XDCZ22.XDCLINK` and *ditto*`.XDCLINKE` libraries dynamically and activate the linklist changes immediately. Examples:

  ```
  SETPROG APF,ADD,DSN=DBCOLE.XDCZ22.XDCLINK,VOL=volser
  SETPROG APF,ADD,DSN=DBCOLE.XDCZ22.XDCLINKE,VOL=volser
  SETPROG LNKLST,DEFINE,NAME=TEMP,COPYFROM=CURRENT,NOCHECK
  SETPROG LNKLST,ADD,NAME=TEMP,DSN=DBCOLE.XDCZ22.XDCLINKE,ATTOP
  ```

```
SETPROG LNKLST,ADD,NAME=TEMP,DSN=DBCOLE.XDCZ22.XDCLINK,ATTOP
SETPROG LNKLST,ACTIVATE,NAME=TEMP
DISPLAY PROG,LNKLST,NAME=CURRENT
F       LLA,REFRESH
```

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

Leaving XDC Out of the Linklist

If you elect to leave the DBCOLE.XDCZ22.XDCLINK and DBCOLE.XDCZ22.XDCLINKE libraries and modules out of the linklist entirely:

- The DBCOLE.XDCZ22.XDCLINK and DBCOLE.XDCZ22.XDCLINKE libraries must be made authorized by adding their dsnames and locations (*volser*) to the APF section of an active PROG*xx* member of the PARMLIB libraries.
- You may issue a SETPROG APF command to activate the authorization immediately without requiring an IPL.
- Users will have to use //JOBLIB or //STEPLIB (or equivalents) in order to use z/XDC. But be aware that when they want to use z/XDC authorized, then the //JOBLIB or //STEPLIB concatenation must NOT include nonauthorized libraries. (Doing so "poisons" the concatenation, causing all libraries to be treated as being nonauthorized.)
- Also note, if you need to run z/XDC authorized under ISPF, then you will NOT be able to use ISPF's LIBDEF facility to access the DBCOLE.XDCZ22.XDCLINK and DBCOLE.XDCZ22.XDCPLPA libraries. This is because LIBDEF does not support authorized libraries.

Making XDCCALLA and XDCCMDA APF-Authorized

XDCCALL**A** and XDCCMD**A** are authorized copies of the (nonauthorized) XDCCALL and XDCCMD load modules located in the DBCOLE.XDCZ22.XDCLINK library. They can be used both as TSO commands and as batch programs (// EXEC PGM=XDCCALLA, for example). They allow users to debug authorized programs and TSO Command Processors. These modules have been created with the "AC=1" attribute and will need to reside in an APF-authorized library..

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

- You must also add their names to the AUTHCMD parameter in an active IKJTSO*xx* member of the PARMLIB libraries.
- Then in order to activate the new AUTHCMD list, you must use TSO's PARMLIB command to cause TSO to refresh its AUTHCMD list in storage.

For details, see **Figure 2** on page 6. For more complete information, see IBM's **Initialization and Tuning Reference** manual, the **TSO/E Customization** manual and the **RACF Command Language Reference** manual.

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

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

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

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

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

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

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

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

Figure 2 Adding XDCCALLA and XDCCMDA to IKJTSO*xx*

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

If you do not wish to permit authorized debugging, then you can create security system rules to prohibit it. See **Defining z/XDC to Your Computer's Security System** on page 12 for more information.

Adding XDCCLIST

The DBCOLE.XDCZ22.XDCCLIST library contains two clists that are used in an ISPF interface to z/XDC. They are named XDCCLIST and.

- **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.XDCZ22.XDCMLIB, *ditto*.XDCPLIB, *ditto*.XDCTLIB and *ditto*.XDCCLIST libraries to your TSO logon

procs.

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

- You may copy both clists to an existing clist library.
- *(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.XDCZ22.XDCCLIST` library to your existing clist library concatenations.

If you copy the `DBCOLE.XDCZ22.XDCCLIST` library members to an existing clist library, then be aware of the following:

- The `DBCOLE.XDCZ22.XDCCLIST` library is distributed with `RECFM=FB` and `LRECL=80`. The clists within it, however, are structured so that they can be copied to a `RECFM=VB` clist library without damage.

- If your target clist library has `RECFM=VB`, then you must use ISPF's Move/Copy Utility (**=3.3**) to perform the copy. `IEBCOPY` cannot do the necessary record format conversion.

- This copying process will have to be repeated every time the factory default instances of the clists are changed by product maintenance.

If you add the `DBCOLE.XDCZ22.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.XDCZ22.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 *//ddname* `INCLUDE` *procname* JCL in your LOGON procs can ease this sort of burden considerably. See **z/OS MVS: JCL Reference** 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.XDCZ22.XDCCLIST` library to the //SYSPROC concatenations.

- It does not matter where, in the //SYSPROC concatenation, the `DBCOLE.XDCZ22.XDCCLIST` library occurs (first, last, middle). Duplicate instances of the clists from other sources are not expected.

- The first library in any concatenation either must have the largest `BLKSIZE` or must specify `DCB=BLKSIZE=`*value* that is as large as or larger than the library having the largest `BLKSIZE`. The `DBCOLE.XDCZ22.XDCCLIST` library's `BLKSIZE` is 27,920.

Adding XDCMLIB, XDCPLIB and XDCTLIB to ISPF

The `DBCOLE.XDCZ22.XDCMLIB`, *ditto*`.XDCPLIB` and *ditto*`.XDCTLIB` libraries contain ISPF panels, messages and table modules (respectively) used for interfacing z/XDC to ISPF. (See **Figure 3** on page 8.) You may either copy these libraries to existing ISPF libraries, or you may add them to your existing library concatenations.

z2.2* Name	z1.13 Name	Type	Purpose
TFSPROF	same	Table	An ISPF profile that causes ISPF to pass all PF-keys and ISPF commands through for processing by z/XDC.
XDCZ22A	XDCZ1DA	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.
XDCZ22B	XDCZ1DB	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 users can more easily invoke z/XDC.
XDCPHELP	same	Panel	Built-in Help for z/XDC's Startup Panel (XDCPANEL).
XDCPHLPx	same	Panel	Various additional Built-in Help panels 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 3 z/XDC's ISPF Clists, Panels, Tables and Message Modules

Alternatively, you may avoid adding the DBCOLE.XDCZ22.XDCMLIB, *ditto*.XDCPLIB and *ditto*.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.*)

Copying z/XDC's ISPF Elements into Existing ISPF Libraries

If you elect to copy DBCOLE.XDCZ22.XDCMLIB, *ditto*.XDCPLIB and *ditto*.XDCTLIB into existing ISPF libraries:

- This copying process will have to be repeated every time the factory default instances of the library members are changed by product maintenance.
- You will have to 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 9 for details.)

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

If you elect to add the DBCOLE.XDCZ22.XDCMLIB, *ditto*.XDCPLIB and *ditto*.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 //*ddname* INCLUDE *procname* JCL in your LOGON procs can ease this sort of burden considerably. See **z/OS MVS: JCL Reference** 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). Duplicate instances of the ISPF elements from other sources are not expected.

- When you add a library to a concatenation, remember that the first library in any concatenation either must have the largest `BLKSIZE` or must specify `DCB BLKSIZE=value` that is as large as or larger than the library having the largest `BLKSIZE`. The z/XDC library `BLKSIZE`s are all 27,920.

- 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 9 for details.)

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

If (*as recommended*) you elect to use the XDCLBDEF clist to dynamically allocate the `DBCOLE.XDCZ22.XDCMLIB`, *ditto*`.XDCPLIB`, *ditto*`.XDCTLIB` and *ditto*`.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.XDCZ22.XDCMLIB`, `DBCOLE.XDCZ22.XDCPLIB`, `DBCOLE.XDCZ22.XDCTLIB` and `DBCOLE.XDCZ22.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.XDCZ22.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 9 for details.) Alternatively, the user can simply run the XDCLBDEF clist from ISPF's Command Shell (**=6**).

Setting Up the XDCPANEL Panel or the XDCLBDEF Clist

The XDCPANEL panel provides an interface allowing ISPF users a quick and easy way to invoke z/XDC to debug their programs. To make it available to your users, you need to choose an invoking panel (such as ISR@PRIM or ISRUTIL) and then modify it as shown in **Figure 4** on page 10.

XDCPANEL can be invoked from an another panel (such as ISRUTIL) either directly [`PANEL(XDCPANEL)`] or indirectly [`CMD(%XDCLBDEF)`]. See **Choosing Between Installing XDCPANEL Directly or Indirectly via XDCLBDEF** on page 10 for important considerations.

In order to install XDCPANEL, see **Figure 4** on page 10 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"[7].

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

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

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

Figure 4 ISPF Panel Mods for Invoking the XDCPANEL Panel

- A line needs to be added (see **Figure 4** above) 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.XDCZ22.XDCMLIB`, *ditto*`.XDCPLIB`, *ditto*`.XDCTLIB` and *ditto*`.XDCCLIST` libraries must first be copied to or concatenated to your //ISPPLIB, //ISPMLIB, //ISPTLIB and //SYSPROC libraries, as discussed above in **Adding XDCMLIB, XDCPLIB and XDCTLIB to ISPF** on page 7 and in **Adding XDCCLIST** on page 6.

- z/XDC's `DBCOLE.XDCZ22.XDCLINK`, *ditto*`.XDCLINKE` and *ditto*`.XDCPLPA` libraries must be copied to or concatenated to your //STEPLIB, PLPA and/or linklist libraries, as discussed above in **Adding the XDCLINK and XDCLINKE Libraries to Your Linklist** on page 3 and **Adding XDCPLPA Load Modules to Common Storage** on page 2.

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

- You need not copy or add the `DBCOLE.XDCZ22.XDCMLIB`, *ditto*`.XDCPLIB`, *ditto*`.XDCTLIB` and *ditto*`.XDCCLIST` libraries to your ISPF libraries. They will be dynamically allocated by XDCLBDEF.

- But you still have to copy `DBCOLE.XDCZ22.XDCCLIST(XDCLBDEF)` library to a //SYSPROC library.

- For **non-authorized** debugging, you need not add the `DBCOLE.XDCZ22.XDCLINK`, *ditto*`.XDCLINKE` and *ditto*`.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.XDCZ22.XDCLINK`, *ditto*`.XDCLINKE` and *ditto*`.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 linklist libraries (as discussed in **Adding the XDCLINK and XDCLINKE Libraries to Your Linklist** on page 3 and **Adding XDCPLPA Load Modules to Common Storage** on page 2).

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

Adding the XDCCMDS Scripts Library

The DBCOLE.XDCZ22.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 12), be sure to give your users **READ** access to this library.

Step 2: ReIPL (Maybe) - A Checklist

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

- When you added the DBCOLE.XDCZ22.XDCPLPA library (or its contents) to the LPA-list, did you also issue the following command?

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

  If not, then do it now or reIPL.

- If you added the DBCOLE.XDCZ22.XDCLINK and DBCOLE.XDCZ22.XDCLINKE libraries to the linklist, did you also issue the following commands?

  ```
  SETPROG APF,ADD,DSN=DBCOLE.XDCZ22.XDCLINK,VOL=volser
  SETPROG APF,ADD,DSN=DBCOLE.XDCZ22.XDCLINKE,VOL=volser
  SETPROG LNKLST,DEFINE,NAME=TEMP,COPYFROM=CURRENT,NOCHECK
  SETPROG LNKLST,ADD,NAME=TEMP,DSN=DBCOLE.XDCZ22.XDCLINKE,ATTOP
  SETPROG LNKLST,ADD,NAME=TEMP,DSN=DBCOLE.XDCZ22.XDCLINK,ATTOP
  SETPROG LNKLST,ACTIVATE,NAME=TEMP
  F       LLA,REFRESH
  ```

  If not, then do it now or reIPL.

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

  ```
  F LLA,REFRESH
  ```

  If not, then do it now.

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

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

  If not, then do it now or reIPL.

- If you added the DBCOLE.XDCZ22.XDCLINK and/or *ditto*.XDCLINKE and/or *ditto*.XDCPLPA library to the APF section of a PROG*xx* member of a PARMLIB library, did you also issue the following commands?

  ```
  SETPROG APF,ADD,DSN=DBCOLE.XDCZ22.XDCLINK,VOL=volser
  SETPROG APF,ADD,DSN=DBCOLE.XDCZ22.XDCLINKE,VOL=volser
  SETPROG APF,ADD,DSN=DBCOLE.XDCZ22.XDCPLPA,VOL=volser
  ```

  If not, then do it now or reIPL.

- If you added the XDCCALLA and XDCCMDA names to the AUTHCMD list of an active IKJTSO*xx* member

of the PARMLIB libraries, did you also issue the following TSO command?

`PARMLIB UPDATE(xx)`

If not, then do it now or reIPL.

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

Step 3: Defining Your License to Use z/XDC

Before you can use z/XDC, you must define to it your license to use it on your current system. The License Definition check is invoked by z/XDC automatically whenever a new debugging session is started. So nobody will be able to use z/XDC until the license is applied.

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

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

Done? Good. Let's continue...

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

Once you have successfully defined your license to z/XDC, you will be able to run debugging sessions. If at some future time you need to display the License Definition Panel, you can do so by starting a normal z/XDC debugging session (such as "`XDCCALL IEFBR14`") and then issuing z/XDC's `LICENSE` command.

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

z/XDC is a generalized debugging tool that can be a powerful aid to any assembler language programmer. z/XDC, in and of itself, is not "authorized"; accordingly, z/XDC both can and should be made available to a computer center's general TSO user population. In a secured computer center, a general user cannot violate system security and integrity with z/XDC.

There are, however, certain capabilities that an installation may wish to restrict the use of to only certain trusted individuals:

- 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 (security permitting) 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.

- 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.

- When a user logs onto z/XDC's Cross Domain Facility (cdf/XDC), system security is called to verify that user's identity and again to determine which background jobs or 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 services that can be set up for z/XDC users and how to establish those services.

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 z2.2 User Guide** or z/XDC's Built-in Help. The title of the discussion is "Help Security". To read it, start a z/XDC debugging session[8], 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`"[9].

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[10] 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.`*description*" (where "*description*" more specifically describes the storage to be altered). Note, this check will be made even for zaps that also require the check for UPDATE access to the "`XDC.FASM.`*jobname*" resource. (For more information, see `HELP SECURITY ZAP` either in z/XDC's Built-in Help or in **z/XDC z2.2 User Guide**.)

5.) When a user signs into the Cross Domain Facility from an idle VTAM terminal, cdf/XDC calls security to verify that user's TSO id and password. (When a user signs onto cdf/XDC from a TSO session, no password check

[8] From TSO's `READY` prompt or from ISPF's Command Shell (**=6**) just type "`XDCCALL IEFBR14`".

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

[10] 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.

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/XDC selects a background job or task to debug, if that job's ownerid does not match the user's userid, then cdf/XDC calls security to see if that user is authorized to debug the selected job or task. It checks for "`UPDATE`" access to the appropriate "`XDC.FASM.`*jobname*" resource.

Important: For more detailed descriptions of z/XDC's resource access control rules, see the `HELP SECURITY` topic in **z/XDC z2.2 User Guide** or in z/XDC's Built-in Help.

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 set up security definitions[11]. 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 5 on page 15 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.

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

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

*"*Jobname*" 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.*jobname*" rule, still also require permission under an appropriate "XDC.ZAP.*description*" rule.

***All entity values start with "XDC" regardless of z/XDC's name. (See **Renaming z/XDC** on page 31 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 5 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 31 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 access 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** and to its **Command Language Reference**.

Also, for a comprehensive discussion about creating the XDC.AUTH, XDC.ZAP.*description* and XDC.FASM.*jobname* rules and about how z/XDC uses them, see the HELP SECURITY topic in **z/XDC z2.2 User Guide** or in z/XDC's Built-in Help.

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

Figure 6 on page 17 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.[12]
- An "XDCSRVER" logonid needs to be created for the Cross Domain Facility. This id must be assigned at least the following attributes: MUSASS, STC and RESTRICT.

[12] 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 "**Renaming z/XDC**" on page 31).

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

*"*Jobname*" 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.*jobname*" rule, still also require permission under an appropriate "XDC.ZAP.*description*" 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 31 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 6 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 CA-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.*description*) TYPE(XDC)
  Controls who can zap what storage. See the HELP SECURITY topic in **z/XDC z2.2 User Guide** or in z/XDC's Built-in Help for details concerning "*description*".

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

  $KEY(FASM.*jobname*) TYPE(XDC)
  UID(...) SERVICE(UPDATE)
  Controls who can have **ZAP** access and cdf/XDC access to the foreign address space named "*jobname*".

- CA-ACF2 permits $KEY values to contain "wildcard" characters (asterisks), thus it is possible, for example, to create one "FASM.*jobname*" rule to control accesses to multiple address spaces. (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 commands (both authorized and nonauthorized). If present, then that table needs to have entries added to it for XDCCALL, XDCCMD, XDCCALLA, XDCCMDA, XDCTFS and XDC.

Finally, in order to run the Cross Domain Facility (cdf/XDC) subserver task (see **Installing z/XDC's Server Job** on ?), you will need to create a logonid for it with at least the following attributes: MUSASS, STC and RESTRICT. The name of this logonid should be "XDCSRVER", and that name must match the name of the proc used to start server/XDC. You also must give the XDCSRVER logonid **READ** access to the AUTH rule.

See CA-ACF2's **Administrator's Guide** for more information about defining generalized resource rules, about logonids and about modifying the control records. Also, DBCOLE.XDCZ22.XDCSAMP(ACF2SAMP) contains several examples of various z/XDC type generalized resource rules.

Also, for a comprehensive discussion about creating the AUTH, ZAP.*description* and FASM.*jobname* rules and about how z/XDC uses them, see the HELP SECURITY topic in **z/XDC z2.2 User Guide** or in z/XDC's Built-in Help.

Using z/XDC in CA-Top Secret Protected Systems (CA-TSS)

Figure 7 on page 19 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 CA-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 CA-TSS's startup parms):

  TSS MODIFY FAC(USERx=NAME=XDCCDF)
  TSS MODIFY FAC(XDCCDF=PGM=XDCSERVE,NOABEND,TENV[13],MULTIUSER,NONPWR,NOTSOC)

[13] 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.

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

*"*Jobname*" 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.*jobname*" rule, still also require permission under an appropriate "XDC.ZAP.*description*" rule.

***The Resource Class of "XDC" is used for security calls regardless of z/XDC's name. (See **Renaming z/XDC** on page 31 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 7 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 Domain Facility's startup proc:
  ```
  TSS   CREATE(XDCCDF)   NAME('name')   DEPT(ownerid)   PASS(NOPW)   FAC(STC,TSO)
        MASTFAC(XDCCDF)
  ```
- Assign XDCSRVER's startup proc to use the XDCCDF ACID:
  ```
  TSS ADD(STC) PROC(XDCSRVER)[14] ACID(XDCCDF)
  ```
- Allow users to logon to XDCCDF:
  ```
  TSS ADD(userid) FAC(XDCCDF)
  ```

[14] In TSS release 4.3 and in TSS release 4.2 with fix #552 applied, the program executed by the XDCSRVER 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 XDCCDF PROC must match the "PGM=XDCSRVER" parameter in the "FACILITY" definition.

- 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 CA-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 CA-TSS, then you will have to apply the CA-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 CA-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/XDC access to JES2's private areas:

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

Note that with 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 CA-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.*description* and FASM.*jobname* rules and about how z/XDC uses them, see the HELP SECURITY topic in **z/XDC z2.2 User Guide** or in z/XDC's Built-in Help.

Step 5: c/XDC Requires an OMVS Segment

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

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

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

Example: If you already know Unix System Services and are comfortable with the whole concept of "OMVS Segments", then just skip this example. You already know what you need to do.

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

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

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

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

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

(This notifies RACF to enable default OMVS processing.)

Done. You can now move on.

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

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

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

z/XDC's System Interface need not, and in fact cannot, be installed by normal methods. Instead, it must be dynamically installed by the server/XDC job. (See **Installing z/XDC's Server Job** on page 22.)

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

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

- It installs (or maybe reinstalls, if necessary) its hook SVC and its service SVC.
- The SVC numbers usually will be 199 and 198, respectively, but that depends upon whether or not those slots in the system's SVC table are empty. If they are not, then other numbers will be chosen.
- The installation process builds for z/XDC a subsystem control table (SSCT) named "XDC*c*" and uses it, among other things, as an anchor for z/XDC's System Interface. It also is a place to save z/XDC's SVC numbers.
- Note, the "*c*" in "XDC*c*" is a release specific special character. For release z2.2, "*c*" is a period ("."), so z2.2's SSCT name is "XDC.". Notes:
    - This intentionally makes it impossible for z/XDC's SSCTs to ever conflict with SSCTs that are installed by z/OS, system products and other vendor products.

    - It also makes it possible for z/XDC itself to recognize all its SSCTs, regardless of clone name (see next bullet).
- When z/XDC is installed as a renamed clone, the "XDC" part of the SSCT's name will be changed to match the clone's name. This allows multiple XDCs to have wholly independent System Interfaces. See **Creating a Renamed Clone of z/XDC** on page 31 for more information.
- The installation process also installs various other system routines to support:
    - Deferred breakpoints (see the `HELP BREAKPOINTS DEFERRED` topic in **z/XDC z2.2 User Guide** or in z/XDC's Built-in Help)
    - Dynamic hooks (`HELP HOOKS DYNAMIC`)
    - And certain end-of-task and end-of-memory cleanup functions.

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

About z/XDC's SVCs

In the case of the two SVCs, z/XDC automatically selects two available SVC numbers (199 and 198 or lower[15]), and it assigns them to the SVCs. Usually:

- The legacy hook SVC will be assigned as SVC 199,
- And the service SVC will be assigned as SVC 198
- **But this is not guaranteed**!

Step 7: Installing z/XDC's Server Job

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

- The ability (known as cdf/XDC) to connect debugging sessions to batch jobs
- The ability to set hooks for creating on-the-fly debugging sessions
- The ability to connect a debugging session across a sysplex to a job running on another system
- The ability to use c/XDC to debug C, C++, Metal C and other C languages.[16]

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

Bouncing server/XDC - Is it Safe?

Yes. It is safe.

Whenever **server/XDC** starts up, it checks to see if z/XDC's System Interface is both installed and current. If it is not, then it will be reinstalled as necessary. Consequently, whenever you apply a maintenance update to z/XDC you will have to bounce **server/XDC**. So not only is it safe, it also sometimes is necessary.

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

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

[16] c/XDC supports debugging a wide variety of C languages, including XL C and C++, SPC C, Metal C, Dignus Systems/C and C++, XL-clang C and C++ and others.

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

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

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

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

Setting up cdf/XDC

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

```
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 8 Typical XDCCDF member to be added to a VTAMLST library for cdf/XDC

1.) DBCOLE.XDCZ22.XDCSRVER(XSRVVTAM) contains a a factory default set of VTAM statements that define a major node to be named "**XDCCDF**" (See **Figure 8** above). This major node is needed by the cdf/XDC subserver. These statements need to be copied to member XDCCDF of one of your system's VTAMLST libraries.

2.) Note, XSRVVTAM is good "right out of the box". No changes are necessary. However, if you do want to change the names of the APPLID's defined to cdf/XDC, then be sure to follow carefully the instructions given in the commentary contained within XSRVVTAM.

3.) If you want the XDCCDF node to come up at VTAM startup time, then add XDCCDF's name to the list of major node names found in an active ATCCON*xx* member of the VTAMLST libraries. Otherwise, the node will have to be started manually by an operator command ("`V NET,ID=XDCCDF,ACT`").

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

Figure 9 Typical `SYSIN` File Parameters for cdf/XDC

4.) `DBCOLE.XDCZ22.XDCSRVER(XSRVPARM)` contains control parameters required by the cdf/XDC subserver (see **Figure 9** above). They tell cdf/XDC what the names are of the various APPLID groups that are defined in your `VTAMLST(XDCCDF)` file. Notes:

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

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

Figure 10 Model Startup JCL for server/XDC

5.) A startup proc for the XDCSRVER started task, such as is shown in **Figure 10** above, needs to be added to one of your system's PROCLIBs. The proc's name typically is "XDCSRVER". A model for the proc can be found in `DBCOLE.XDCZ22.XDCSRVER(XSRVPROC)`. The JCL should be edited as indicated in **Figure 10**.

6.) server/XDC needs to have **READ** access to the `XDC.AUTH`[17] or `AUTH`[18] security rule. Please read the `HELP SECURITY AUTH` topic in **z/XDC z2.2 User Guide** or in z/XDC's Built-in Help for more information.

7.) There are two ways that users can connect to batch job debugging sessions:

- One is via a z/XDC Startup Panel in ISPF.
- The other is via a direct logon from an idle 3270 type terminal.

If you chose to support direct logons, then you need to create RACF, CA-ACF2 or CA-Top Secret security definitions that support such logons. XDCSRVER needs to be defined to your security system such that it will permit the cdf/XDC subserver to make security calls to it in behalf of other users. Please refer back to **"Defining z/XDC to Your Computer's Security System"** on page 12 for details.

[17] For RACF

[18] For CA-ACF2 or CA-Top Secret

8.) In addition, when a user connects into cdf/XDC looking for a debugging session to connect to, cdf/XDC has to examine all pending sessions, to see which one(s) the user has the right to connect to. For this purpose, cdf/XDC issues "RACHECK" type security calls to make this determination. This check is done before cdf/XDC displays to the user the list of jobs pending debugging. Therefore, if a pending job is not displayed that should be displayed, then this is an indication that the appropriate security system rules are not set up correctly.

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

The rule that cdf/XDC checks is:

- "FASM.*jobname*" for CA-ACF2 and CA-Top Secret systems,
- "XDC.FASM.*jobname*" for RACF systems.

Refer back to **Defining z/XDC to Your Computer's Security System** on page 12 for more information.

9.) The following operator command need to be added to an active COMMND*xx* member of your PARMLIB libraries:

```
COM='S XDCSRVER'
```

This command starts server/XDC. If the server is started too soon (i.e. before the XDCCDF VTAM node comes up), then the cdf/XDC subserver will just wait for VTAM to come up.

10.) When server/XDC successfully completes its initialization, it will issue the following two messages and then wait for work to do:

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

Step 8: The Installation Verification Test

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

Method 1: Navigate to z/XDC's Startup Panel, fill it in as shown in Figure 11 on page 26, 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 31), then the first thing you have to do is type z/XDC's new name (Z22, for example) on the command line and press **ENTER**. This will automagically reconfigure the panel to use the renamed z/XDC.

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

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

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

Figure 11 z/XDC's Startup Panel in ISPF

Method 2: Navigate to ISPF's "Command Shell" (=6) and type XDCLBDEF:
This will work if you have suitably modified and installed the XDCLBDEF clist. If it does work, then you will see the Startup Panel shown in **Figure 11** on page 26.

Method 3: Navigate to ISPF's "Command Shell" or to TSO's READY prompt and type: `XDCCALL IEFBR14`
This will work if you have installed z/XDC's load libraries (`DBCOLE.XDCZ22.XDCLINK`, *ditto*.XDCLINKE and *ditto*.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 12** on page 26 or **Figure 13** on page 27 to appear at your terminal.

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

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

.  DBC830I XDC z2.2 ENTERED NONAUTHORIZED, AMODE(24), UNDER RB#3 FROM TCB#9 IN
.  DBC830I DBCOLEB (ASN=0095.17)
.  DBC891I XDC z2.2 ENTERED AS AN ESTAI OWNED BY TCB#8
.  DBC831I THE ERROR LEVEL AND RETRY LEVEL ENVIRONMENTS ARE THE SAME
_  DBC841I DEAD MSG: 2711 *** DBC901I SESSION IS READY FOR DEBUGGING, TYPE H AT
                     THE LEFT FOR MORE INFORMATION ***



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

Figure 12 Initial Screen Displayed by z/XDC on a Classic 43x80 3270 Terminal

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

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

.  DBC830I XDC z2.2 ENTERED NONAUTHORIZED, AMODE(31), UNDER RB#3 FROM TCB#10 IN
.  DBC830I DBCOLE7 (ASN=008B.23)
.  DBC891I XDC z2.2 ENTERED AS AN ESTAI OWNED BY TCB#9
.  DBC831I THE ERROR LEVEL AND RETRY LEVEL ENVIRONMENTS ARE THE SAME
_  DBC841I DEAD MSG: 2711 *** DBC901I SESSION IS READY FOR DEBUGGING, TYPE H AT THE LEFT FOR MORE
                      INFORMATION ***


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

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

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

Figure 13 Initial Screen Displayed by z/XDC on a Terminal Set to Large Dimensions

If, however, you see a display such as that shown in **Figure 14** on page 27, 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 9 for more information.

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

```
TCB#9 RB#1 ------------------ z/XDC TPUT INTERFACE ---------------------------
XDC ===>
   ISPP100  Panel 'XDCZ22A' error
   ISPP100  PANEL NOT FOUND.

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

Figure 14 Error When z/XDC is Not Properly Installed into ISPF

In any case, once you see the display in **Figure 12** or **Figure 13**, it is verified that z/XDC is installed in the proper system libraries and that it is operational. You may proceed, if you wish, to try out the various z/XDC commands.

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

Testing Authorized Debugging

If you have installed the authorized commands 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 15 on page 28, and press ENTER.** (In particular, **before** pressing ENTER, be sure to respond YES to the question: "**Should CMD/PGM start APF auth?**") This will work if you have installed z/XDC's Startup Panel into properly.

- **Or go to ISPF's "Command Shell" or to TSO's READY prompt and type:** XDCCALLA IEFBR14
  This will work if you have installed z/XDC's load libraries (DBCOLE.XDCZ22.XDCLINK, DBCOLE.XDCZ22.XDCLINKE and DBCOLE.XDCZ22.XDCPLPA) or load modules into the system's search order for your TSO session, and also added XDCCALLA (and XDCCMDA) to an active IKJTSO*xx* member of the PARMLIB libraries.

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

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

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

Figure 15 Starting z/XDC Authorized from its Startup Panel in ISPF

Doing either of these things should cause the fullscreen display shown in **Figure 16** on page 29 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 an active IKJTSO*xx* member of the PARMLIB libraries. Please see **Making XDCCALLA and XDCCMDA APF-Authorized**, on page 5, for more information.

```
TCB#6 RB#1 ------------------------------------------------ z/XDC TPUT INTERFACE
----------------------------------------
XDC ===>
.  DBC854I z/XDC for z/OS
.  DBC855I COPYRIGHT (C) COLESOFT PARTNERS, INC. - 2010-2016. ALL RIGHTS RESERVED
.  DBC856I FOR PRODUCT SUPPORT, PLEASE CALL COLESOFT AT 540-456-8210

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

.  DBC830I XDC z2.2 ENTERED *AUTHORIZED*, AMODE(31), UNDER RB#3 FROM TCB#6 IN
.  DBC830I DBCOLE7 (ASN=008B.23)
.  DBC891I XDC z2.2 ENTERED AS AN ESTAI OWNED BY TCB#5
.  DBC831I THE ERROR LEVEL AND RETRY LEVEL ENVIRONMENTS ARE THE SAME
_  DBC841I DEAD MSG: 2711 *** DBC901I SESSION IS READY FOR DEBUGGING, TYPE H AT THE LEFT FOR MORE
                      INFORMATION ***


XDC ===> L PSWE;L EPSWE;L BEA;;L RWREGS;;L AREGS
_    PSWE 07851000 80000000 00000000_000ACE90 (cc-LO) (31) - DBCCALL.XDCCALL.XDCCALL
_  EPSWE 07851000 80000000 00000000_0009D9AA (cc-LO) (31) - DBC810W NOT WITHIN ANY IDENTIFIABLE MODULE OR ANY OTHER OBJECT
_  Breaking Event Address (BEA): XDCCALLA+273E

_     RW0  00000000_00000000 00000000_000B0F90  FFFFFFFF_E7C4C3C3 FFFFFFFF_C1D3D340  *.....................XDCC....ALL *
_     RW4  FFFFFFFF_C4C2C3C3 FFFFFFFF_C1D3D340  FFFFFFFF_8457204E FFFFFFFF_0457304D  *....DBCC....ALL ....d..+.......(*
_     RW8  FFFFFFFF_0077FAB8 FFFFFFFF_000977E0  FFFFFFFF_0457404C FFFFFFFF_0457504B  *..............\...... <......&.*
_     RW12 FFFFFFFF_007B0E18 00000000_0009D610  00000000_8009DA26 00000000_800ACE90  *.....#........o.................*

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

Figure 16 Initial Screen Displayed by Authorized Mode z/XDC (on a Large Geometry Terminal)

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, SWAP and *=jump* commands while debugging authorized programs running in TSO.
- This restriction does not exist when using cdf/XDC 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 **Installing z/XDC's Server Job** on page 22, for more information.

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

Step 9: cdf/XDC Installation Verification

In order to test cdf/XDC, you first must have a background job that has abended and is asking for cdf/XDC services. The job shown in **Figure 17** on page 29 will serve this purpose.

```
//      --- JOB ---
//IVP      EXEC PGM=XDCIVP
//STEPLIB   DD  DISP=SHR,DSN=DBCOLE.XDCZ22.XDCLINK
//          DD  DISP=SHR,DSN=DBCOLE.XDCZ22.XDCLINKE
//          DD  DISP=SHR,DSN=DBCOLE.XDCZ22.XDCPLPA
//XDCPROF   DD  DISP=SHR,DSN=Your ISPF profile library          [optional]
```

Figure 17 Model JCL for Testing cdf/XDC

The //XDCPROF DD card, if present, is used by z/XDC to find the user's profile data (member XDCDPZ22, assuming it exists). From the profile, z/XDC determines such things as how long it should wait for a user to log onto the debugging session and whether or not it should display a `WTOR` to the operators permitting them to abort the logon wait prior to its timing out.

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

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

- TSO users with ISPF can use z/XDC's Startup Panel ("XDCPANEL", see **Adding XDCMLIB, XDCPLIB and XDCTLIB** on page 7). From that panel, select option **3** to connect to cdf/XDC.
- Other users can simply logon directly to cdf/XDC from an idle VTAM terminal. By default, the necessary command is "`LOGON APPLID=`*name*".[19] [20] cdf/XDC will then present the user with a screen requesting his TSO userid and password (**Figure 18** on page 30).

```
TCB#23 RB#1 -----------------XDC-CDF VTAM INTERFACE ------------ A.S. XDCCDF
XDC ===>

   XX    XX   DDDDDDD     CCCCCC            /    CCCCCC    DDDDDDD    FFFFFFFF
   XX    XX   DD    DD   CC    CC          //   CC    CC   DD    DD   FF
    XX  XX    DD    DD   CC               //    CC         DD    DD   FF
     XXXX     DD    DD   CC              //     CC         DD    DD   FF
      XX      DD    DD   CC             //      CC         DD    DD   FFFFFFF
     XXXX     DD    DD   CC            //       CC         DD    DD   FF
    XX  XX    DD    DD   CC           //        CC         DD    DD   FF
   XX    XX   DD    DD   CC    CC    //         CC    CC   DD    DD   FF
   XX    XX   DDDDDDD     CCCCCC     /           CCCCCC    DDDDDDD    FF




   USERID        ==>
   PASSWORD      ==>
   NEW PASSWORD ==>
   RACF GROUP    ==>

   PROGRAMMERS NAME        ==>
   ACCOUNTING INFORMATION ==>
```

Figure 18 Logon Screen for VTAM Connections to z/XDC's Cross Domain Facility

Once within cdf/XDC, the user will be presented with a list of abended background jobs and system tasks that he is permitted (according to system security) to debug (**Figure 19** on page 31). 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.

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

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

```
TCB#23 RB#1 -----------------XDC-CDF VTAM INTERFACE ------------ A.S. XDCCDF
XDC ===>

   Type S below at the left to select the debugging session to connect to.
   JOBNAME   STEPNAME  PROCSTEP  PROGRAM   MODULE                ASID  OWNERID
_  CDFXDCTS  A                   XDCCALL   WTOR                  0015  DBCOLE
```

Figure 19 cdf/XDC Job Selection menu

For further information, though, the user should type HELP CDF[21] before proceeding.

Step 10: Creating a Renamed Clone of z/XDC (for Coexistence with multiple XDCs)

If you want to install z/XDC z2.2 into your system and have it coexist with a different version or release of XDC, you can do so, but in order to do so, you will have to rename one or the other of the XDCs. (Usually, one would decide to rename the newer release of z/XDC or a beta version of z/XDC, perhaps for a "try out" period.) You may rename z/XDC to any other 3-character name you choose. One such name might be **Z22**. (But my personal favorite is **DBC**.)

The names of all z/XDC load modules in DBCOLE.XDCZ22.XDCLINK, *ditto*.XDCLINKE and *ditto*.XDCPLPA start with the characters **XDC**. These first three characters can be changed to Z22, to your initials or to any other legal three characters you choose. The remaining characters in each name must not be altered.

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

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

- If you change the names of any z/XDC load modules, then you must change the names of all of them in all three load libraries: DBCOLE.XDCZ22.XDCLINK, *ditto*.XDCLINKE and *ditto*.XDCPLPA.
- The z/XDC load modules may be renamed directly within the Product Libraries; but remember, in order to apply maintenance, you will have to rename them all back to **XDC***name*, then apply the maintenance, and then rename them all back to their clone names.
- You will have to repeat this process each time that you apply z/XDC maintenance.
- If user programs are written to invoke z/XDC internally, then they may become dependent upon the name that you choose. Thus, your name choice may become "locked in" over time.

Considerations for Using a Renamed z/XDC

The ability to create renamed clones of z/XDC makes it possible to simultaneously run any combination of z/XDC versions and releases that you wish. However, there are several considerations that need to be taken into account. (For the purposes of these illustration, let me presume that you have chosen to create a clone named of **Z22**.)

[21] However, you cannot issue this HELP CDF command until after selecting the job to be debugged, not before.

- Users will have to invoke z/XDC using clone names (**Z22***name* names) instead of **XDC***name* names. Example:
  ```
  //A EXEC PGM=Z22CALLA,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'Z22'
  ```

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

  My personal favorite way to do this would be to scan the TIOT for a ddname of the form //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: `//XDCISZ22 DD DUMMY`. For more information, see `HELP XDCCLONES #XDCIS`.

- Any //XDC*whatever* DD cards or dynamic allocations you might have for your debugging sessions (profiles, traces, other controls) will have to be changed. Examples:
  ```
  //Z22QUICK DD DUMMY
  //Z22PROF  DD DISP=SHR,DSN=userid.ISP.ISPPROF
  ```

- At IPL time, you will need to run a Z22SRVER job similar to the XDCSRVER job described in "**Installing z/XDC's Server Job**" on page 22.

- In ISPF, to use the **Z22** name, on the XDC Startup Panel, you simply have to type `Z22` on the command line and then press **ENTER**. The panel will automatically adjust itself to invoked **Z22** instead of XDC.

- You would have to create a Z22CDF VTAM node similar to that described "**Setting up cdf/XDC**" on page 23. Briefly, you would need to do the following:

  - Make a copy of your XDCSRVER startup proc, and rename both it and everything within it from `XDC`*whatever* to `Z22`*whatever*.

  - Make a copy of the //SYSIN file referenced by your **Z22**SRVER startup proc, and perform similar renames within it.

  - Make a copy of your XDCCDF VTAMLST member, rename it to **Z22**CDF and perform similar renames within it.

  - For auto-start of **Z22**CDF's nodes at IPL time, add **Z22**CDF to an active ATCCON*xx* member of the VTAMLST libraries.

  - Also, add a `COM='S Z22SRVER'` command to an active COMMND*xx* member of the PARMLIB libraries.

- For TSO, you will have to add **Z22**CALLA and **Z22**CMDA to the AUTHCMD section of an active IKJTSO*xx* member of the PARMLIB libraries.

- Note, security system rules do **not** have to be specifically created or changed for any clone. All clones of z/XDC will honor security rules based upon the name **XDC** (not the clone's name).

- However, for sites that use the 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 **Z22**CALL, **Z22**CMD, **Z22**CALLA, **Z22**CMDA, **Z22**TFS and **Z22** itself.

- An **Z22**SRVER logonid needs to be created for the Cross Domain Facility. This id, like the XDCSRVER id, must be assigned at least the following attributes: `MUSASS`, `STC` and `RESTRICT`.

- z/XDC Profiles: When a user first starts up **Z22**, if he had a saved profile in the older XDC, then **Z22** will not be able to find it automatically. So to access his old profile, a user will have to issue the following commands:

```
PROFILE READ DPZ22 XDC
PROFILE SAVE
```

  The `PROFILE SAVE` command will cause the profile to be rewritten using the name **Z22**DPZ22 (z/XDC's default profile name when running under the clone name of Z22).

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

Step 11: Making DBCOLE.XDCZ22.XDCADATA the Default MAPLIB Library

The `DBCOLE.XDCZ22.XDCADATA` library contains ADATA[22] for a few of z/XDC's load modules, the most important of which is the XDCMAPS load module. XDCMAPS contains mapping information (SYM data) for a very large number of z/OS system control blocks. The `DBCOLE.XDCZ22.XDCADATA` library contains corresponding source image information (ADATA) for those same control blocks. Consequently, it is useful to set up `DBCOLE.XDCZ22.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 MAPLIBS list that will be automatically available to a user's debugging session without requiring him to issue the `SET MAPLIBS` command. This default MAPLIB information is kept in the user's session profile.

The `DBCOLE.XDCZ22.XDCADATA` library (or whatever you have renamed it to) is a good candidate for being a default MAPLIB library. To accomplish this, you will have to start a z/XDC debugging session and then issue the following commands:

```
SET MAPLIBS RESET DBCOLE.XDCZ22.XDCADATA[23] SAVE
PROFILE SAVE
```

But don't actually issue these commands yet. Read the next section ("**Creating a Default Profile: TFSDPZ22**") first.

Step 12: Creating a Default Profile: TFSDPZ22

TFSDPZ22, when it exists, is a system-wide default profile for z/XDC (see **Adding XDCMLIB, XDCPLIB and XDCTLIB to ISPF** on page 7). When a user invokes z/XDC for the first time, if a TFSDPZ22 member exists in an ISPF Table Library (//ISPTLIB), then that is loaded for his use.[24] It is loaded by z/XDC regardless of whether z/XDC is running within or outside of an ISPF environment.

---

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

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

[24] If a user has previously saved his own z/XDC profiles, then z/XDC will not automatically load TFSDPZ22 Instead, it will load the user's profile and then automatically convert it, if necessary, to z2.2 format. But z/XDC will not write the converted profile back to disk unless and until the user issues a `PROFILE SAVE` command.

If a user changes his profile data and then issues a `PROFILE SAVE XDC` command, then z/XDC stores the changed profile into the user's own profile library in a member named ***xxx***DPZ22, where "***xxx***" matches z/XDC's name (see **Renaming z/XDC** on page 31).

Subsequently, the next time this user starts z/XDC, his ***xxx***DPZ22 will be loaded instead of the System-wide TFSDPZ22.

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

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

When a System-wide default profile (TFSDPZ22) does exist, it will be used automatically only when the user has no current or older default profile of his own.

When a System-wide default profile (TFSDPZ22) does **not** exist, then z/XDC will automatically use one of its own built-in Factory Default profiles. (See `HELP PROFILES FACTORYDEFAULTS` for details.)

So if the factory default profile settings are OK as they are, then there is no reason for you to go to the trouble of creating a local version. However, it is likely that there are several factory default settings your programmers will not like. For detailed suggestions about what you should consider changing, see **Which System-Wide Default Profile Values You Might Consider Changing** on page 35.

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

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

If you wish to change any of the factory default settings, then you will have to create a System-wide default profile for use at your Data Center. See **Which System-Wide Default Profile Values You Might Consider Changing** (on page 35) for suggestions about which profile settings you might consider changing first.

To create a local System-wide default profile, proceed as follows:

- Start up z/XDC. (From ISPF's Command Shell [=6], issue `XDCCALL IEFBR14`.)
- Issue `PROFILE RESET` *[name]* to insure the desired set of Factory Defaults is loaded. "*[name]*" can be one of `AWIDE`, `A80`, `CWIDE`, `C80`, `DWIDE` or `D80`. (See `HELP COMMANDS PROFILE RESET` for more information.)
- Issue `SET MAPLIBS RESET DBCOLE.XDCZ22.XDCADATA`[25] `SAVE` to make the `DBCOLE.XDCZ22.XDCADATA` library the default MAPLIB library for future debugging sessions. (See the preceding section, **Making `DBCOLE.XDCZ22.XDCADATA` the Default MAPLIB Library**, page 33, for more information.)
- Issue `PROFILE` to start the Profile Menuing System (see **Figure 20** on page 35).

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

- One by one, select (with an `S` at the left) each of the menu's panels and make those changes you deem necessary.
- Use **PF3** to close each menu panel and move on to the next.
- Use **PF3** again to exit the Profile menuing System altogether.
- Type `PROFILE SAVE XDC TFS` to save the changed profile back into your profile library under the member named TFSDPZ22.
- Leave z/XDC and invoke ISPF's "Move/Copy Utility" (`=3.3`).
- Move (not just copy) your changed profile (member name TFSDPZ22) from your profile library to your system's ISPF table library.

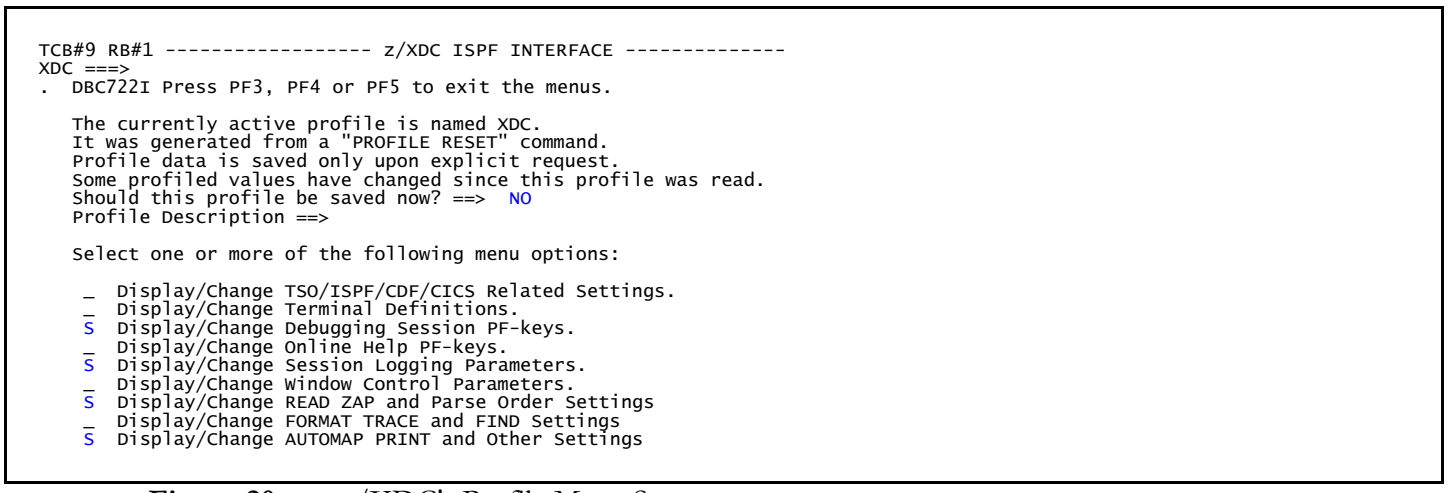

Figure 20 z/XDC's Profile Menu System

Which System-Wide Default Profile Values You Might Consider Changing

As discussed above, it is important to set up the `DBCOLE.XDCZ22.XDCADATA` library as a default MAPLIB library for use by your user community. If you haven't done so already, then please see **Making `DBCOLE.XDCZ22.XDCADATA` the Default MAPLIB Library** on page 33 for more information.

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

- **Individual PF-key Assignments:**
  Many of z/XDC's factory default PF-key assignments are "real nonstandard". (See **Figure 21** on page 36.) Accordingly, the temptation, for many people, is high to change them, but please think twice before doing so.

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

```
TCB#6 RB#1 ------------------ z/XDC TPUT INTERFACE --------------------------
XDC ===>
.  DBC722I Press PF3 to accept changes and return. Press PF5 to accept changes
           and exit the menus.
.  DBC722I Press PF4 to discard changes and return.

    _  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 ===>
.  DBC722I Press PF3 to accept changes and return. Press PF5 to accept changes
           and exit the menus.
.  DBC722I Press PF4 to discard changes and return.

    _  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 21 Default PF-key Definitions

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

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

- **Session Log File Allocation:**
  z/XDC supports automatic or manual logging of debugging sessions either to disk or SYSOUT spool files. The initial default is for z/XDC to automatically log all debugging sessions to "HOLD" class X on spool. If "X" is not your standard held output class, then you may change the default to an any class you prefer. (See **Figure 22** on page 37.)

```
TCB#6 RB#1 ------------------ z/XDC TPUT INTERFACE --------------------------
XDC ===>
.  DBC722I Press PF3 to accept changes and return. Press PF5 to accept changes
           and exit the menus.
.  DBC722I Press PF4 to discard changes and return.

   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 22 Profile Settings for z/XDC's Session Log

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

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

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

- **Changing the Default Scripts Library (used by z/XDC's READ command):**
  The default profiles all set the default scripts library to DBCOLE.XDCZ22.XDCCMDS. But if you have installed z/XDC's Product Libraries using a different HLQ (High Level Qualifier), then you will want to change the scripts library name in your system-wide default profiles. The Profile Menu panel for doing this is shown in **Figure 23** on page 37. Simply overtype DBCOLE.XDCZ22.XDCCMDS with the actual name of the scripts library you want to use.

```
TCB#6 RB#1 ------------------ z/XDC TPUT INTERFACE --------------------------
XDC ===>
.  DBC722I Press PF3 to accept changes and return. Press PF5 to accept changes
           and exit the menus.
.  DBC722I Press PF4 to discard changes and return.

           READ, ZAP and Parse Order Settings              CHOICES
    NO     Allow Zapping of Store Protected Storage?       YES    NO
    YES    Limit Mnemonic Zaps by Instruction Length?      YES    NO
    YES    Echo Scripted Cmds and Displays as They Happen? YES    NO
           Default Scripts Library Name DBCOLE.XDCZ22.XDCCMDS
    notset Script File Sequence# Field?                    PRESENT ABSENT DETECT

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

Figure 23 Profile Settings for READ, ZAP and PARSEORDER

- **//XDCPRINT File Characteristics:**
  z/XDC supports a PRINT command that users can use to print various things such as selected portions of the Built-in Help and even entire z/XDC manuals. Users can use the Profile Menu System to customize the characteristics of the printed output file. (See **Figure 24** on page 38.) You may want to check out these settings and perhaps change them if necessary to match the characteristics of your most typical printers.

```
TCB#7 RB#1 (AUTH) (DBC496W!) ---------------------------------------- z/XDC TPUT INTERFACE -------
XDC ===>
.  DBC722I Press PF3 to accept changes and return. Press PF5 to accept changes and exit the menus.
.  DBC722I Press PF4 to discard changes and return.

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

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

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

Figure 24 Profile Settings for AUTOMAP, PRINT and Misc

Refer to the HELP FULLSCREEN PROFILE topic (in **z/XDC z2.2 User Guide** or in z/XDC's Built-in Help) for more detailed information about user profiles.

Step 13: Exit Routines

If you or your users have exit routines that have been written for prior versions or releases of z/XDC, then you probably should reassemble them. There may have been some changes in the programming interface that the exits use. See **z/XDC z2.2 Release Guide** for more information. Also, check out the commentary located in the #DBCPARM macro found in the DBCOLE.XDCZ22.XDCMACS library.

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

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

For more information about **User Exits**, please read the HELP EXITS topic in **z/XDC z2.2 User Guide** or in z/XDC's Built-in 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. But with a UCI exit, user communications can be directed towards any device or path desired. For more information see the HELP EXITS UCI topic in **z/XDC z2.2 User Guide** or in z/XDC's Built-in Help.

  Source code for a sample UCI exit can be found in DBCOLE.XDCZ22.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" or "EXRL" of a user SVC). Sometimes, user SVC routines make non-standard returns to locations other than the immediately following instruction. This exit can be used to predict for z/XDC when a user SVC will make such a non-standard return and where that return will be made to. This exit is necessary during z/XDC tracing in order for z/XDC not to lose control of the trace.

  - **A "User Commands" Exit**
    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.XDCZ22.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 z2.2 User Guide** or in z/XDC's Built-in Help.

  Source for a sample router module can be found in `DBCOLE.XDCZ22.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 z2.2 User Guide** or in z/XDC's Built-in Help for more information.