ColeSoft®

# XDCMMEF

## The XDC Module Mapping Extraction Facility

## User's Guide

www.colesoft.com

# Table of Contents

Revision date: 20th August, 2021 - RWS

# Executive Overview

XDCMMEF is a free-ware utility provided by ColeSoft Marketing, Inc., makers of the z/XDC family of debugging tools.

z/XDC and its various Features allow software engineers to more easily perfect the applications that they offer to their user communities. These applications are written in Assembler or C language, and are made available as the production software that is licensed to you.

While every effort is made to ensure that ISV software products are error-free, problems may be encountered during execution or testing, and sometimes dumps are generated. Previously, the only recourse the customer had was to send a dump to the vendor for analysis. This has now changed.

ColeSoft Marketing has released dump/XDC, which extends z/XDC's considerable power into the realm of dump-reading. However, a dump does not always contain some information that dump/XDC needs in order to help your vendor to address your problem.

ColeSoft Marketing, Inc. provides XDCMMEF, the "XDC Module Mapping Extraction Facility". This entirely benign utility extracts the information that your vendor needs. Such information includes load module maps, which contain NO confidential information, but help dump/XDC to do its job.

Prior to XDCMMEF sending load modules to a vendor could create a potential security exposure. It would also be impractical due to the immensity of the information to be transmitted. Instead a software engineer might have travelled to the customer's site in order to conduct analysis. This involved additional expense, often deprived the software engineer of all the resources necessary AND created a potential security exposure in itself – for both parties.

XDCMMEF solves this problem by extracting ONLY non-secure information from load libraries for off-site examination by vendor engineers. It creates NO security exposure.

You are welcome to contact ColeSoft Marketing, Inc. with any questions regarding the use of XDCMMEF. We may be reached at support@colesoft.com or via telephone at (540) 456-8210.

# Drilling Down

One of z/XDC's most powerful features is the ability to map storage with source image and symbol data from the original program being debugged. However, when an ISV receives a dump from a downstream customer, he does not receive copies of all load modules represented in the dump. This makes it impossible for dump/XDC to accurately map the modules in the dump. This is where the XDCMMEF program steps in.

XDCMMEF is a standalone utility that can scan multiple sets of load libraries and extract just the mapping information out to a compressed, portable dataset. If an ISV sends XDCMMEF to its customers (or asks the customer to download the program from the COLESOFT web site), then they can return to the ISV mapping information for any or all modules represented in a dump.

The extracted information may include:

- The system nucleus;
- Programs loaded in the PLPA;
- Programs located in the LINKLIST, JOBLIB and STEPLIB;
- A module's ESD map (the layout of all the CSECTs within the module);
- Optionally, any SYM data records the module may contain (symbols from assemblies);
- Optionally, any ADATA the module may contain (source code information from assemblies).

The extracted information does not contain:

- Any program code;
- Any program data.

In other words, the extracted data does not contain any part of a load module that actually gets loaded into storage and executed.

XDCMMEF's output dataset is compressed. It can safely be FTP'd in binary (image) mode.

XDCMMEF allows a downstream customer to send an ISV necessary debugging data without also sending private, licensed, proprietary and bulky load modules. So, privacy is retained, and the volume of data shipped is substantially reduced.

## SYM data and ADATA

The XDCMMEF program allows the user to control whether or not SYM data and ADATA are offloaded by the program.

XDCMMEF does not, by default, allow the capture of ADATA Type 30 records (source records). It may be configured to do so by explicit action of the user. However, the transmission of ADATA and/or SYM data between parties may create an exposure of proprietary information. If relevant, both parties to such transmission should ensure that a non-Disclosure Agreement is in place between them.

***ColeSoft Marketing, Inc. assumes no liability with regard to the transmission of ADATA between third parties.***

SYM data and ADATA extraction are controlled by 2 sets of parameters. This allows a coarse decision as to whether SYM data and/or ADATA is produced at all, and a fine decision as to which source of SYM data and/or ADATA is to be captured. In the ADATA case, you may decide separately whether ADATA source records are offloaded at all.

> The NOSYM parameter is a default that suppresses all SYM data. The [NO]SYMPLPA, [NO]SYMLL, and [NO]SYMLIB parameters can be used to control the source(s) of offloaded SYM data. By default, only SYMLIB is in effect.
>
> The NOADATA parameter is a default that suppresses all ADATA. The [NO]ADATALL and [NO]ADATALIB parameters can be used to control the source(s) of offloaded ADATA. By default, only ADATALIB is in effect.

Also, in the case of ADATA, ADATA source records are only offloaded if the ADSRC parameter is in effect. By default, this parameter is NOT in effect. Also note that if NONAMES is in effect (suppression of names in the offload file), ADATA source records are not offloaded.

## XDCMMEF's Capabilities

In a single execution, XDCMMEF can collect information about:

- The system nucleus;
- The PLPA (information about the FLPA, MPLA, and DLPA IS NOT extracted);
- The linklist;
- Up to 50 load library concatenations. Each concatenation can have from 1 to 255 load libraries (PDS) and/or program object libraries (PDSE) specified.

## What is Extracted

XDCMMEF extracts information about:

- The load module or program object attributes, (such as length and flags);
- Records appropriate to the format (load module or program object) that are used by z/XDC to emulate either reading a load module or the Binder API;
- SYM and ADATA (optional);
- Optional records that describe the internal format, including:
  - Class names;
  - A list of section and entry names offsets, and lengths;
  - Translator information;
  - IDR information.

## Output Format

XDCMMEF's output is a compressed, sequential file. While it may contain many records, it will be significantly smaller than the datasets/members that it describes.

## System Requirements

- This program must execute in z/OS 1.6 or later.
- It requires a z9 machine or later.
- It can run in any execution environment under z/OS.

# Sample JCL

Here is some sample JCL that can be used to execute the program in batch:

```
//EXTRACT  EXEC PGM=XDCMMEF,PARM='<parameters>'
//STEPLIB  DD  DISP=SHR,DSN=<library containing the XDCMMEF program>
//SYSPRINT DD  SYSOUT=*
//OUTFILE  DD  DISP=(NEW,CATLG),
//             DSN=<output file name>,
//             UNIT=3390,VOL=SER=<serial>,
//             SPACE=(CYL,(50,10),RLSE)
//* DDnames for LIBn as required.
```

If the XDCMMEF program is loaded to the linklist or LPA, there will be no need for the STEPLIB DD statement.

XDCMMEF normally executes in batch, but it can also be used under TSO by using the CALL command.

# Parameters

The XDCMMEF program takes a standard OS parameter string. Its default parameters are normally sufficient. If you need to alter XDCMMEF's parameters, then parameter values in the string can be separated by blanks and/or commas. Leading and trailing blanks are ignored, as are null or blank parameters.

If conflicting parameters are specified, the LAST one stated takes effect. Each group of parameters is shown. Several parameters will force other parameter values which are indicated below. All forcing will occur after all parameter processing is complete.

If you need to provide a significant number of parameters to the program, and the list exceeds the z/OS JCL parm field limit of 100 characters, and you are executing the XDCMMEF program on z/OS 2.1 or later, then you may use the PARMDD facility to pass a very long parameter string to XDCMMEF. In this case the IGNORE parameter may be useful at the end of your parameter list, so that you can have each parameter followed by a comma, on each line of the parameter file with the IGNORE parameter on the last line.

Here is a list of the parameters accepted by XDCMMEF. Some parameters are covered in more detail elsewhere in the document. Default values are underlined. Each group of parameters is shown together. Again, if more than one parameter in a group is specified, the last one that appears in the parameter string takes effect.

## Selection

- NUC – indicates that nucleus information is to be extracted.
- NONUC – indicates that nucleus information is not to be extracted.

- PLPA – indicates that PLPA information is to be extracted.
- NOPLPA – indicates the PLPA information is not to be extracted.

- LL – indicates the linklist information is to be extracted.
- NOLL – indicates that linklist information is not to be extracted.

- SYS – the same as specifying all of NUC, PLPA, and LL
- NOSYS – the same as specifying all of NONUC, NOPLPA, and NOLL

- LIB – indicates that library information is to be extracted.
- NOLIB – indicates that library information is not to be extracted.

If NONUC, NOPLPA, NOLL, and NOLIB are in effect, an error message will be generated, as no actions were specified at all.

## Compression

- OPTCOMP – indicates that output file compression is optional and compression may not be possible if the CSRCMPSC service is not available on the local installation of z/OS.
- NOCOMP – indicates that no output file compression will be done.
- YESCOMP – indicates that output file compression will always be done. If a particular compression technique is not possible in the environment, an error message will be generated.

- COMP1 – compression technique 1 to be used if compressing (IBM LZ using a dictionary generated by ColeSoft). Specification of this technique and YESCOMP will cause an error if the CSRCMPSC service is not available on this release of z/OS. Specification of this technique and OPTCOMP will permit no compression if the hardware CMPSC instruction is not available.
- COMP2 – compression technique 2 is an XDC internal compression technique. Note that OPTCOMP will be treated as YESCOMP for this technique.

## Reporting

- STDRPT – indicates that a standard report is to be written. The standard report will show the dataset name and the number of members in each dataset in total, and the number of members selected in each dataset.
- DETRPT – indicates that a detailed report is to be written. The detailed report has everything in the standard report, and also has a detailed list of members showing their status.
- NORPT – indicates that no report is to be written. Information about XDCMMEF's execution, the environment, what is unloaded, and the processing of each unload phase will still be written.

## SYM data

- NOSYM – indicates that no SYM information is to be written at all. All the NOSYMxxx parameters are treated as specified.
- SYM – indicates that SYM information is to be written based on the SYMxxx parameter below.
- ALLSYM – indicates that SYM information is to be forced. All the SYMxxx parameters are treated as specified.

- SYMPLPA – If a load module in the PLPA has SYM data, it may be written.

- NOSYMPLPA – no PLPA module SYM data will be written.

- SYMLL – if a load module or program object in the linklist has SYM data, it may be written.
- NOSYMLL - no linklist module SYM data will be written.

- SYMLIB – if a load module or program object in a LIB ddname concatenation has SYM data, it may be written.
- NOSYMLIB – no LIB module SYM data will be written.

## ADATA

- NOADATA – indicates that no ADATA information is to be written at all. All the NOADATAxxx parameters are treated as specified.
- ADATA – indicates that ADATA information is to be written Based on the ADATAxxx parameters, below. But note that the relevant ADATAxxx parameter must also be specified to actually have ADATA written.
- ALLADATA – indicates that ADATA information is to be forced. All the ADATAxxx parameters are treated as specified.
- ADATALL – if a program object in the linklist has ADATA, it may be written.
- NOADATALL – no linklist program object ADATA will be written.
- ADATALIB – if a program object in a LIBx ddname has ADATA, it may be written .
- NOADATALIB – no LIBx program object ADATA will be written.
- ADSRC – indicates that ADATA source records (ADATA record type 30) will be written if ADATA is being written (provided that NONAMES is not in effect).
- NOADSRC – indicates that ADATA source records will NOT be written even if ADATA is being written.

## Miscellaneous

- NAMES – indicates that dataset names (and other names) are to be written.
- NONAMES – indicates that dataset names and other names are not to be written ('?' used as the dataset name or other name instead). This security feature prevents actual dataset names (etc.) from appearing in the output file. Also, ADATA Type 30 (source records) will NOT be written even if the ADATA parameter is specified. Note that regardless of the use of NONAMES, member names are always written, as they are used to identify specific members and are needed when the information is being used. The use of the NONAMES parameter will reduce the utility of XDC-MMEF data.

- IGNORE – this parameter does nothing. It may be useful as the last parameter when using a PARMDD file, because you don't need to follow it with a comma.

### *Examples*

Some parameter examples are presented.

**<nothing>**

This is the most common parameter setting, Indicated in the JCL by either not specifying the PARM

parameter at all, or specifying “PARM=’ ’” (i.e. a blank string)

In this case, all of the default values (as indicated above) are in effect. Information will be extracted from the Nucleus, PLPA, linklist, and any LIBn ddnames.

**PARM=’NOSYS’**

This parameter setting extracts information from any LIBn ddnames only. This setting may be used if you need to extract information from more than 50 LIBn ddnames.

**PARM=’ALLSYM,ALLADATA’**

This parameter allows SYM data and ADATA to be extracted (if available).

# DDnames

Here is a list of the DDnames that this program uses.

**SYSPRINT**

This is the report file. This file is required.
It is RECFM=VBA, LRECL=137. If not specified, the BLKSIZE will default (to whatever value the system chooses).

**OUTFILE**

This is the output dataset. It is required. It must be a sequential file.
- The RECFM is FB.
- The LRECL can be any value from 80 to the BLKSIZE. If left as 0, the program itself will choose a suitable LRECL.
- BLKSIZE will be determined by the system, but is normally ½ of a track. On a 3390, this will be 27998 or the closest integer multiple of the LRECL that is less than this.
- Regardless of the LRECL or BLKSIZE chosen, the file records that the program writes will be spanned across logical dataset records.

# Compression

If compression is active (as controlled by the YESCOMP, NOCOMP and OPTCOMP parameters), the output file will be compressed.

**LIBn**

These are input library concatenations that are used if the default LIB parameter is in effect. ”n” can be from 1 to 50 without a leading zero. If the NOLIB parameter is in effect, then LIBn ddnames are ignored.
Each unique LIBn ddname can have from 1 to 255 PDS/PDSE libraries specified (concatenated). Each library in the list must be a load module or program object library.

LIBn ddnames are processed in ”n” order – not allocation order! The first missing LIBn ddname will stop LIBn processing. A LIBn allocation to DUMMY is ignored but processing of the next LIBn will continue.

Extra definitions are automatically produced for all uniquely-named concatenated libraries.

# Allocation Limits

The number of datasets that can be allocated in one JCL step is limited by the size of the z/OS TIOT as set by the ALLOCxx parmlib parameter during system IPL. This may be a problem if a large number of LIB datasets need to be described.

The maximum TIOT size of 64K allows for 3273 datasets in a job step. If the TIOT size is 32K, you can have up to 1635 datasets defined. Other TIOT sizes result in other proportional limits being imposed.

The standard datasets needed by XDCMMEF (SYSPRINT and OUTFILE, and maybe STEPLIB) means that the program always uses 2 or 3 TIOT slots. Of the remaining slots:

- Each library in a LIBn ddname takes 1 TIOT slot. So, if a LIBn description has 8 datasets concatenated, then 8 TIOT slots are used.
- The XDCMMEF program makes use of dynamic allocation. If XDCMMEF executes under z/OS 1.13 or later, AND the option in the DEVSUPxx parmlib member NON_VSAM_XTIOT=YES is in effect then an extended TIOT will be used for all dynamic allocations, and the dynamic allocations will have no effect on the limits.
- If the environment is NOT at least z/OS 1.13 OR the option NON_VSAM_XTIOT is not in effect, dynamic allocations by the program are also counted towards that limit. These allocations are in effect for the life of the program, and reduce the number of available datasets in the following ways:
- If the NUC parameter is in effect, there is no effect;

  If the PLPA parameter is in effect, all datasets in the PLPA concatenation (as described in the parmlib member LPALSTxx) will be dynamically allocated, and hence the number of datasets that can be specified in the JCL will be reduced by that amount;

  If the LL parameter is in effect, all datasets in the currently in-effect linklist (as described in the parmlib member LNKLSTxx or PROGxx) will be dynamically allocated, and hence the number of datasets that can be specified in the JCL will be reduced by that amount;

  Each LIBn dataset will be dynamically allocated. That means that the number of datasets defined in LIBn dd statements has to be counted twice when calculating the number used.

If these limits are exceeded, then you may execute the XDCMMEF program several times. In one execution you may just extract information about the Nucleus, PLPA, and Linklist and have no LIBn ddnames (so no LIBn information is extracted)). In subsequent executions, use a parm that has ”NOSYS” and add the relevant LIBn DDnames.
Send all the resulting output datasets separately. **They cannot be concatenated together.**

# Return Codes

The program will give the following return codes (unless it ABENDs, which is covered below):
00 The program has executed normally.
04 The program executed, but at least one warning has been issued.
08 (or greater) If the return code is 8 or greater, then a significant error occurred.

# ABEND codes

Under a set of restricted circumstances, XDCMMEF could ABEND. These are listed below:

## *U0001 User ABEND*

A reason code will indicate the specific ABEND reason. Note that if the operating system does not support an ABEND reason, the reason code will be in R15.
The reason code must be used to determine the specific problem. When this ABEND is issued, it is not possible to write more detailed information.

- Reason code 00000001. This indicates that the O/S is a lower level than z/OS 1.6. See “System Requirements” above.
- Reason code 00000002. This indicates that a specific hardware facility is not available. R2 will have the bit number (from 0 to n) indicating the specific hardware facility that was missing. The bit number can be looked up in the “facility indications” section of Chapter 4 of the Principles of Operation.
- Reason code 0000003. The program was unable to obtain storage for its main work area. R2 will have the return code from the STORAGE OBTAIN macro.
- Reason code 0000004. The SYSPRINT file (the report file) could not be opened.
- Reason code 0000005. A DCB ABEND occurred on the report file. R2 will contain the DCB ABEND code and reason code. The format of the information in the fullword R2 is X‘AAA000cc’ where ”AAA” is the system abend code, ”0” is binary 0, and ”cc” is the ABEND reason code.
- Reason code 0000006. An I/O error occurred in the report file.

## *S0C1 System ABEND*

If the program encounters an illogical condition during execution, it will ABEND by executing the string X’00DEAD’. This will result in the PSW pointing to the X’AD’ in the string. The next byte will have the length of a text string, describing the reason for the failure.

In the majority of cases, this ABEND will be captured by the program, and it will print the ABEND information, and then end with a return code of 16.

If any ABEND occurs in XDCMMEF, please contact ColeSoft Marketing, Inc for assistance. You may contact us by email at support@colesoft.com. You may also reach us by telephone at (540) 456-8210.

# Extraction of System Libraries

## Nucleus extraction

If the NUC parameter is in effect, then:

- The Nucleus suffix will be written. A single character will be appended to ”IEANUC0” to obtain the actual nucleus name (typically ”1” – leading to ”IEANUC01”);
- The nucleus map will be used to generate a list of CSECTs and labels that make up the nucleus, along with their offsets and lengths;
- A set of output information will be generated. The output includes a set of constructed CESD records, as if the nucleus was a load module. Unlike the member(s) in SYS1.NUCLEUS, however, the constructed CESD reflects the actual position of sections in memory (taking into account residence options and read-only status, for example).

## PLPA extraction

If the PLPA parameter is in effect, then the current system PLPA information for the current IPL is written. DLPA (Dynamic LPA) or MLPA (Modified LPA) or FLPA (Fixed LPA) information is NOT extracted because this information is not kept by the system. The LPALST is used to obtain a list of the datasets that make up the PLPA concatenation.

The list of datasets is treated like a LIBn list, in that:

- Each dataset is dynamically allocated;
- The directory of each dataset (PDS) is read and a member list for the dataset is built;
- A composite member list is built, containing unique member names. Where a member name occurs more than once, only the first member (in dataset order) is used;
- Each unique member will result in a set of output information being generated,
- PLPA datasets will always be load libraries (PDSs). This is a z/OS restriction.

## Linklist extraction

If the LL parameter is in effect, then the current system linklist information is written.

**Note:** It is possible to alter the Linklist after IPL. The actual Linklist extracted is the one assigned to the region where the program is executing, or the current system linklist at the time that the XDCMMEF program executes. This is why we strongly urge that the XDCMMEF program be run during the same IPL as when the original dump was generated – the information will all match nicely.

Processing is as follows:

- Either the region’s assigned Linklist (if a specific one is there) or the system Linklist will be used. A specific region-assigned linklist will only be different from the system’s linklist if it is changed after the region starts.
- The list of Linklist datasets is treated like a LIBn list, in that:
- Each dataset is dynamically allocated;
- The directory of each dataset (PDS/PDSE) is read and a member list for the dataset is built;
- A composite member list is built, containing unique member names. Where a member name oc-

curs more than once, only the first member (in dataset order) is used;
- Each unique member will result in a set of output information being generated;
- Linklist libraries can be PDSs or PDSEs. If PDSEs, they contain program objects –XDCMMEF automatically handles this.

  However, the only way for XDCMMEF to obtain program object information is to use the “fast data” Binder API.

  While faster than the ”standard” Binder API, the ”fast data” API is still much slower than reading a load module directly.

  However, the format of a program object is not documented, and you must use the IBM-supplied binder APIs to obtain information about the internal format of a program object.

## LIBx extraction

The list of datasets concatenated to the ‘LIBx’ ddname (x is 1 to 50) is processed as follows:

- Each dataset is dynamically allocated;
- The directory of each dataset (PDS/PDSE) is read and a member list for the dataset is built;
- A composite member list is built, only containing unique member names. Where a member name occurs more than once, only the first member (in dataset order) is used;
- Each member will result in a set of output information being generated;
- LIBn libraries can be PDSs or PDSEs. The same comments about the requirement to use the IBM-supplied binder API as mentioned above applies.

## ADATA notes

ADATA is extracted from program objects and written to the XDCMMEF output file only if the NOADATA option is NOT in effect, and one of the ADATAxxx (ADATALL or ADATALIB) is in effect.

The following ADATA record types are extracted:

0000 – JOB information.
0002 – translation unit start/end information.
0020 – ESD information.
0030 – source records (maybe, see below).
0034 - DC/DS information.
0036 - machine instruction information.
0042 – symbol information.

Other ADATA record types are not extracted.

## ADATA source records (type 0030)

Even if extracting ADATA, source records (ADATA record type 0030) are only extracted if:’

- The ADSRC parameter is in effect (this parameter, by default is NOT in effect).
- The NAMES parameter is in effect (this parameter, by default is in effect).

# How to make XDCMMEF's contents available to dump/XDC

The output of XDCMMEF should be transmitted to the people who will perform problem determination on the original dump. They will find this information quite valuable. Thanks for your time and effort in getting it to them.

On the recipient's side:

Each unique output file from XDCMMEF can be described to the dump/XDC IPCS verb exit using the IPCS literal equate "DXDCUMMEFDSNn".

Alternatively, the output of the XDCMMEF job can be linked to a dump with the dump/XDC "DEFINE MMEFDSN <dataname> command.

Decompression is handled transparently. If the particular compression technique used by the XDCMMEF program is not supported by the release of z/XDC or if the z/OS system does not support the compression technique used, an error message will be generated.

The XDCMMEF dataset must be catalogued. The IPCS verb exit will check that the dataset and will read the header records to verify that it is an XDCMMEF output dataset for the correct CPU and operating system.

Both the IPCS verb exit and z/XDC will detect errors in the FTP process.

When the dump/XDC IPCS verb exit passes control to z/XDC, the IPCS literal names (as described above) are used to generate DEFINE MMEFDSN commands. Each unique dataset will result in a command being generated.

When asked to map a module in a dump, z/XDC will use from the output of XDCMMEF instead of scanning local load modules and program objects.

# Conclusion

The output of the XDCMMEF utility can be of great help to the people analyzing a dump with the dump/XDC Feature of the z/XDC suite of debugging tools. Now, in addition to the dump itself, the dump/XDC user will be able to increase his or her understanding dramatically, leading to a faster resolution of the problem.

We at Cole Software support our products *enthusiastically*. If you should have any questions, please contact us at the coordinates below:

- Our website is www.colesoft.com.
- First-level support: Bob Shimizu at (800) XDC-5150, or via e-mail at bob@colesoft.com
- Second-level support: Frank Chu at (540) 456-6164, or via email at frank@colesoft.com
- Both Frank and Dave Cole monitor support@colesoft.com.
- Dave Cole can be reached at dbcole@colesoft.com.