Precompiler for "EXEC DLI" Commands

This chapter covers the following topics:

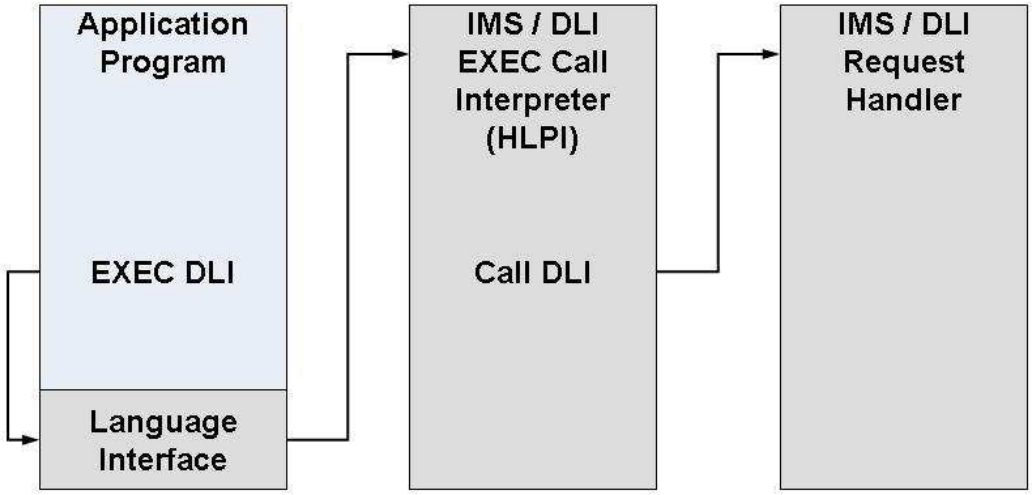
- Introduction
- ADL Precompiler Input
- ADL Precompiler Output
- COBOL Generated Code
- PL/I Generated Code
- CICS Command Language Translator
- Linkage-Editor Requirements for Application Programs
- z/OS JCL Requirements
- z/VSE JCS Requirements

Introduction

This section is only of interest for installations in which application programs using the Higher Level Programming Interface (HLPI) need to be converted. It describes the different possibilities and procedures available for converting such application programs written in COBOL or PL/I .

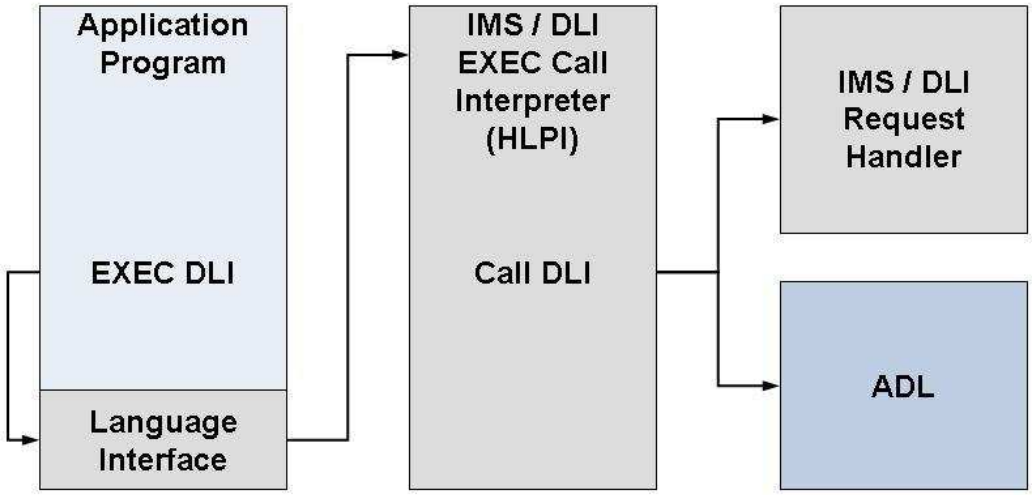
When an application program using the HLPI issues a DL/I request in an unconverted environment (i.e. where ADL is not involved), an extra interface layer is used to interpret the call from the application program (which originates from an `EXEC` command) and transform them into normal calls. The figure below illustrates this process.

HLPI Interface



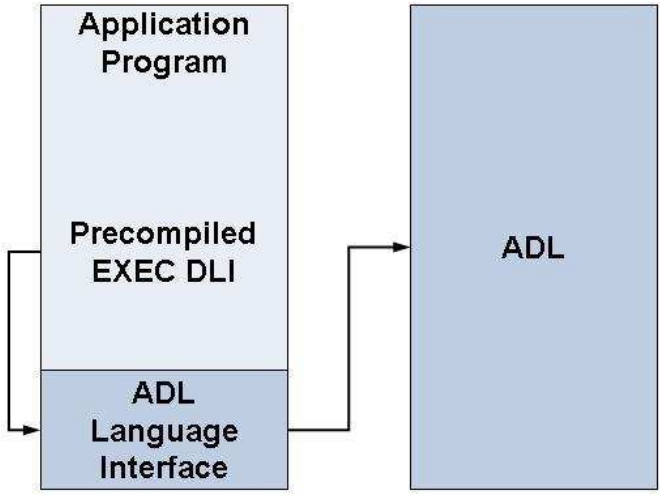
Application programs using the HLPI can be run against ADL in one of two different ways: with or without the ADL precompiler.

Running HLPI Programs without the ADL Precompiler



The application programs do not issue standard DL/I calls directly: instead, these are issued by the HLPI. This means that the DL/I environment must be present. Command level programs can be run against ADL in mixed mode.

Running HLPI Programs with the ADL Precompiler



The ADL precompiler translates each `EXEC DLI` command in the application program into one (and only one) call to ADL. This call is passed directly to ADL in the same manner as for normal DL/I calls. This means that no extra interface layer is required. Thus, in a completely converted environment, DL/I need not still be available. When the ADL precompiler has been used, however, application programs cannot be run in mixed mode.

ADL Precompiler Input

The ADL precompiler accepts as input a source program written in PL/ I or COBOL in which `EXEC DLI` commands have been coded. It produces as output an equivalent source program in which the commands have been translated into statements in the language of the source program. The statements generated result in a call to the ADL nucleus (batch or CICS) to fulfill the function requested.

For more details on the syntax of the `EXEC DLI` commands, see the appropriate IBM documentations.

The ADL precompiler is activated separately before the compile and link edit steps.

The program source input data set must contain fixed length records, which must be 80 bytes long. The first statement of the source program must be a CBL statement for COBOL or a `*PROCESS` statement for PL/ I . This control statement may be passed on unchanged as part of the program source output.

The control statement contains the parameters for the ADL precompiler itself, and has the following syntax:

```
LAN=XXX,keyword=value
```

Keyword	Explanation
INPUT	(z/VSE only). Indicates whether the source program input should be read in from the reader or from disk/tape. Possible values:

Keyword	Explanation
	DISK The program source input is read in from disk or tape. You must specify DAZIN5D or DAZIN5T, respectively, as DTF. The corresponding logical unit is SYS014. This default logical unit can be overwritten by the ADL 'FX' parameter as described in the section *ADL Parameter Module* in the *ADL Installation* documentation. Since the ADL precompiler expects a record size of 80 bytes, the 'FX' setting should be overwritten during the generation of the precompiler nucleus DAZNUCP. For example: FX=(14,80,80) READER The program source input is read in from the reader, i.e. from SYSIPT. Default: READER
LAN	The source program language, a three-character abbreviation. Possible values: CBL for COBOL PLI for PL/I Default: none
LIST	Indicates whether or not the program listing, including the EXEC commands with sequence numbers, is to be produced. Possible values: Y : (the program listing is to be produced) N : (the program listing is not to be produced) Default: Y
MARGIN	(PL/I only) Specifies the left and right margin of the input statements of the source program. The ADL precompiler will only scan that part of any input source statement between the two margins. The syntax of this parameter is as follows: MARGIN=(L,R) L : Left margin of source program Possible values: 1-71 Default: 2 R : Right margin of source program Possible values: 2-72 Default: 72
PREF	The three-character prefix to be used for variables (COBOL) and ENTRY names (PL/I). Possible values: AXX where: A : is any character which may be used as the first character of a COBOL variable name or PL/I ENTRY name. X : any character allowed to be the second or any subsequent character of a COBOL variable name or PL/I ENTRY name. Default: DAZ
QNUM	(COBOL only). The maximum number of qualification statements to be expected in any one command. Possible values: 1-999 Default: 8
SEGM	Indicates whether or not a program segment is to be precompiled. Possible values: Y : A program segment is to be precompiled for later inclusion in the main program. This means that only the EXEC commands will be translated. No variables and no user DIB will be generated. N : A main program is to be precompiled. Default: N If SEGM=Y is specified for a PL/I program segment, make sure that the parameter PREF is specified as well. The ADL precompiler generates a separate ENTRY statement with a unique name for each EXEC command in PL/I. Where a program segment is precompiled separately, the ENTRY names generated must differ from those in the main program. This can be achieved by specifying a different prefix (PREF parameter).
SNUM	(COBOL only) The maximum number of SSAs to be expected in any one EXEC command. Possible values: 1-15 Default: 15

Keyword	Explanation
UDIB	Indicates whether or not fields need to be generated for the user DL/I Interface Block (DIB). Possible values: Y : (fields for the user DIB will be generated). N : (fields for the user DIB will not be generated). Default: N
XOPTS	Specifies whether or not the first statement should be output. Possible values: Y : where the first statement should be output. N : where the first statement should not be output. Default: N

ADL Precompiler Output

The program source output is output as fixed length records with a length of 80 bytes.

The first output listing produced contains messages produced by the ADL precompiler. Where one or more errors were encountered during translation, an error message with the following layout will be produced:

```
Warning nnnn from ADL module DAZEXEC/DAZEXSER at address aaaaa
ADLnnnn message ............
Error occurred during interpretation of EXEC DLI statement with sequence no:
99999
```

where

nnnn	is the number of the message,
aaaaa	is the offset within the ADL module DAZEXEC in which the message was generated, and
99999	is the sequence number of the `EXEC DLI` command which was found to be in error. This sequence number corresponds to the number printed on the output data set containing the complete listing (see below).

The second output listing produced contains the original source program including the `EXEC DLI` commands. In addition, each `EXEC DLI` command has been given a sequence number which can be used to find particular commands found to be in error. Where an `EXEC DLI` command is found to be in error, the error message written to the message output listing is reproduced.

COBOL Generated Code

For COBOL, each `EXEC` command is replaced by a series of `MOVE` statements followed by a `CALL` statement. The `MOVE` statements take care of possible type conversions for numeric arguments and assign constants to data variables. For this purpose declarations for these temporary variables are automatically included in the working storage. Declarations for the user DIB are also automatically included in the working storage. It is possible to precompile program segments separately for later inclusion in a main program.

PL/I Generated Code

For PL/ I each EXEC command is replaced by a DO statement, a declaration of a generated unique ENTRY name, a CALL statement and an END statement. The ENTRY declaration takes care of possible type conversions for numeric arguments. The ADL precompiler generates the declarations for the user DIB variables for each valid PL/I PROCEDURE statement. It is possible to precompile program segments separately for later inclusion in a main program.

CICS Command Language Translator

Where an application program also contains EXEC CICS commands, the CICS Command Language Translator has to process the application program as well before it can be compiled. The CICS Command Language Translator may be activated either before or after the ADL precompiler. In either case, make sure that the translator options for the CICS Command Language Translator do not specify DLI.

Linkage-Editor Requirements for Application Programs

After having passed the ADL precompiler, the CICS language translator and the compiler, the application program must be linked together with the appropriate language interfaces for ADL. Depending on the operational environment, the following modules are to be linked to the application program:

CICS online programs:

z/OS - DAZLICI3

z/VSE - DAZLICID

Batch programs:

z/OS - DAZLIBAT

z/VSE - DAZLIBAT

All language interface modules described above are on the installation tape in the ADL load library. The language interfaces mentioned above replace the IBM modules DFSLI000 (z/OS) and DLZLI000 (z/VSE). For details on how to link-edit command level application programs, see the *CICS Installation and Operations Guide*

Important:
All application programs which have been linked with DAZLICI2 of ADL 2.2 or before, must be relinked with DAZLICI3. DAZLICI3 is CICS release independent.

z/OS JCL Requirements

The table below lists the data sets used by the ADL precompiler during processing of an application program.

DDname	Medium	Description
DAZIN2	Disk/Tape	Program source input
DAZIN1	Reader	ADL precompiler control statement
DAZOUT1	Printer	Program listing
DAZOUT2	Printer	Error messages
DAZOUT4	Disk/Tape	Precompiled program

Example

The following is an example of an ADL Precompiler Run for a COBOL application program:

```
//PRE      EXEC PGM=DAZIFP,PARM='PRE,DAZEXPRE'
 //STEPLIB  DD   DISP=SHR,DSN=ADL.LOAD
 //         DD   DISP=SHR,DSN=ADABAS.LOAD
 //DAZOUT1  DD   SYSOUT=X
 //DAZOUT2  DD   SYSOUT=X
 //DAZOUT4  DD   DSN=&&TEMP,SPACE=(TRK,(20,20)),UNIT=VIO,DISP=(,PASS),
 //         DCB=(RECFM=FB,DSORG=PS,BLKSIZE=3120,LRECL=80)
 //DDCARD   DD   *
 ADARUN PROGRAM=USER,...
 //*
 //DAZIN1   DD   *
 LAN=CBL
 //DAZIN2   DD   *
       ...
       ...   application program source code
       ...
 //*
 //APPLPROG EXEC DFHEITCL
 //TRN.SYSIN DD DSN=&&TEMP,DISP=(OLD,DELETE)
 //LKED.ADL  DD DISP=SHR,DSN=ADLxxx.LOAD
 //LKED.SYSIN DD *
  INCLUDE ADL(DAZLICI3)
  NAME anyname(R)
 /*
```

z/VSE JCS Requirements

The table below lists the data sets used by the ADL precompiler during application program processing.

DTF	Logical Unit	Medium	Description
DAZIN2	SYSIPT	Reader	Program source input *1
DAZIN5D	SYS014	Disk	Program source input *2/ *3
DAZIN5T	SYS014	Tape	Program source input *2/ *3
DAZIN1	SYSIPT	Reader	ADL precompiler control statement
DAZOUT1	SYSLST	Printer	Program listing
DAZOUT2	SYS011	Printer	Error messages
DAZOT3D	SYS013	Disk	Error messages *4
DAZOT3T	SYS013	Tape	Error messages *4
DAZOUT4	SYSxxx	Disk	Precompiled program

*1 Only required, if the default input mode INPUT=READER is active.

*2 Only required, if the input mode `INPUT=DISK` is specified.

*3 Either one (disk or tape) is required. The logical unit indicated, is the default logical unit. To change it, specify the `FX` parameter as described in the section *ADL Parameter Module* in the *ADL Installation* documentation. Note that the ADL precompiler expects a record length of 80 bytes.

*4 Only required when only one logical printer is available. In this case, the message which is normally directed to `DAZOUT2` as a second print file will be written to disk. At the end of the job, it will be read from disk and routed to `DAZOUT1`.

If the default input mode `INPUT=READER` is active, the control input for the batch monitor `(DAZIFP)`, `ADARUN`, the ADL precompiler, and the program source input are all read in from `SYSIPT`. The control statements must be specified in the following order:

```
PRE,DAZEXPRE                                  input for DAZIFP
/*
ADARUN ...                                    input for ADARUN
/*
LAN= ...                                      input for the ADL precompiler
/*
.                                             application program source
.
.
/*
```

Example

The following is an example of an ADL Precompiler Run for a COBOL application program:

```
// ASSGN SYS010,DISK,VOL=volume,SHR
// DLBL DAZOUT4,'punch-dataset',0,SD
// EXTENT SYS010,volume,,,rtrk,ntrks
// ASSGN SYS013,DISK,VOL=volume,SHR
// DLBL DAZOT3D,'temp-dataset',0,SD
// EXTENT SYS013,volume,,,rtrk,ntrks
// ASSGN SYS013,DISK,VOL=volume,SHR
// DLBL DAZIN3D,'temp-dataset',0,SD
// EXEC PROC=ADLLIBS
// EXEC DAZIFP,SIZE=512K
PRE,DAZEXPRE
/*
ADARUN PROGRAM=USER,...
/*
LAN=CBL
/*
  COBOL application program source
  ....
/*
// DLBL IJSYSPH,'cobol-translation',0
// EXTENT SYSPCH,,1,0,rtrk,ntrks
ASSGN SYSPCH,DISK,VOL=volume,SHR
// DLBL IJSYSIN,'punch-dataset',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volume,SHR
// EXEC DFHECP1$
CLOSE SYSIPT,SYSRDR
CLOSE SYSPCH,cuu
// DLBL IJSYSIN,'cobol-translation',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volume,SHR
// OPTION CATAL
 PHASE pgmname,*
 INCLUDE DFHECI
// EXEC FCOBOL
 INCLUDE DAZLICID
/*
CLOSE SYSIPT,SYSRDR
// EXEC LNKEDT
/&
// JOB RESET
ASSGN SYSIPT,SYSRDR
ASSGN SYSPCH,cuu
/&
```