z/OS Installation

This chapter covers the following topics:

- Overview

- Initial Load of the ADL Libraries (Step 1)

- Initial Load of the ADL Directory File (Step 2)

- Initial Program Load of the ADL Natural Programs (Step 3)

- Creating the ADL Parameter Module (Step 4)

- Creating the ADL Executable Load Modules (Steps 5 - 10)

- Creating the Consistency Front-Ends (Steps 11 - 12)

Overview

This chapter describes the steps necessary to install the Adabas Bridge for DL/I (ADL) in a z/OS environment. After performing these steps, you will be able to run the ADL data base conversion utilities, to use the ADL Online Services and to operate the ADL Interfaces for DL/I and Adabas calls.

For easy reference, all installation steps are summarized below. In general, all steps must be performed in all environments. Exceptions are clearly marked as such in the detailed description of the individual steps.

Step	Description
Step 1	Load the ADL libraries to disk.
Step 2	Load the ADL directory file.
Step 3	Load the ADL Online Services and the ADL Installation Verification Package.
Step 4	Create the ADL parameter module.
Step 5	Create the executable ADL data conversion module.
Step 6	Create the executable ADL precompiler module.
Step 7	Create the executable module for the ADL CALLDLI Interface in batch.
Step 8	Create the executable module for the ADL Consistency Interface in batch.
Step 9	Create the executable module for the ADL Interfaces under CICS.
Step 10	Create the executable ADL batch region controller.
Step 11	Create the ADL substitute for the Adabas link module in batch.
Step 12	Link-edit the Adabas link module under CICS with the ADL exit `ADLEXITB`.

Note:
If you use Software AG's System Maintenance Aid (SMA), steps 1 to 9 are performed by SMA.

Initial Load of the ADL Libraries (Step 1)

Load the ADL libraries to disk using the JCL given below as an example.

Library Space Requirements (z/OS)

The table below gives an estimate of how much space is needed for various device types (primary cylinders, directory tracks).

Library	3390
Load	3, 10
Source	6, 36
LC23	1, 2

Copying the Tape Contents to a z/OS Disk

If you are using SMA, refer to the *System Maintenance Aid* documentation (included in the current edition of the Natural documentation CD).

If you are *not* using SMA, follow the instructions below.

This section explains how to:

- Copy dataset `COPY.JOB` from tape to disk.

- Modify this dataset to conform to your local naming conventions.

The JCL in this dataset is then used to copy all datasets from tape to disk.

If the datasets for more than one product are delivered on the tape, the dataset `COPY.JOB` contains the JCL to unload the datasets for all delivered products from the tape to your disk.

After that, you will have to perform the individual install procedure for each component.

- Step 1 - Copy Dataset COPY.JOB from Tape to Disk
- Step 2 - Modify COPY.JOB on Your Disk
- Step 3 - Submit COPY.JOB

Step 1 - Copy Dataset COPY.JOB from Tape to Disk

The dataset `COPY.JOB` (Label 2) contains the JCL to unload all other existing datasets from tape to disk. To unload `COPY.JOB`, use the following sample JCL:

```
//SAGTAPE JOB SAG,CLASS=1,MSGCLASS=X
//* -------------------------------
//COPY EXEC PGM=IEBGENER
//SYSUT1 DD DSN=COPY.JOB,
// DISP=(OLD,PASS),
```

```
// UNIT=(CASS,,DEFER),
// VOL=(,RETAIN,SER=tape-volume),
// LABEL=(2,SL)
//SYSUT2 DD DSN=hilev.COPY.JOB,
// DISP=(NEW,CATLG,DELETE),
// UNIT=3390,VOL=SER=volume,
// SPACE=(TRK,(1,1),RLSE),
// DCB=*.SYSUT1
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//
```

where:

hilev is a valid high level qualifier
tape-volume is the tape volume name, for example: T12345
volume is the disk volume name

Step 2 - Modify COPY.JOB on Your Disk

Modify the `COPY.JOB` on your disk to conform to your local naming conventions and set the disk space parameters before submitting this job:

- Set *HILEV* to a valid high level qualifier.

- Set *LOCATION* to a storage location.

- Set *EXPDT* to a valid expiration date.

Step 3 - Submit COPY.JOB

Submit `COPY.JOB` to unload all other datasets from the tape to your disk.

Initial Load of the ADL Directory File (Step 2)

Load the ADL Directory File. This file is a standard Adabas file unloaded with the `ADAULD` utility of Adabas. You may use the JCL in the ADL source library member `ADLINS2` as an example.

If ADL is already installed at your site, and you want to continue to use the existing ADL directory file, you may simply delete the obsolete error messages from the existing Directory File and load the new messages from the unloaded file provided on the installation tape. You may use the JCL in the ADL source library member `ADLINS2A` as an example.

Initial Program Load of the ADL Natural Programs (Step 3)

Load the ADL Online Services and the `DDM` for the ADL Directory File, `ADB-CONTROL`. The `INPL` file on the installation tape was created with the Natural `SYSOBJH` utility and is compatible with Natural version 4.1 and upwards. You may use the JCL in the ADL source library member `ADLINS3` as an example.

Note that the `INPL` file on the ADL installation tape contains a `DDM` for the ADL Directory File and the object modules of the library `SYSADL` with the ADL-supplied subprograms `ADLERROR` and `ADLACTIV`. These subprograms may be used by Natural applications operating under the ADL Consistency Interface.

The same step loads also the ADL Installation Verification Package (SYSADLIV) and the related DDMs.

Creating the ADL Parameter Module (Step 4)

Create the ADL parameter module (DAZPARM) by assembling the DAZPARM macro. This procedure is common to both z/OS and z/VSE and is described in the chapter *ADL Parameter Module* in this documentation.

You may use the JCL in the ADL Source Library member ADLINS4 as an example.

Creating the ADL Executable Load Modules (Steps 5 - 10)

The following steps are performed:

- Step 5

- Step 6

- Step 7

- Step 8

- Step 9

- Step 10

Step 5

Create the executable ADL CBC utility module by running the link editor and specifying the following link edit directives:

```
ORDER DAZPARM
INCLUDE ADLLOAD(DAZPARM)
INCLUDE ADLLOAD(DAZCCGEN)
INCLUDE ADLLOAD(DAZCCOUT)
INCLUDE ADLLOAD(DAZCCSUB)
INCLUDE ADLLOAD(DAZAXES)
INCLUDE ADLLOAD(DAZSERV)
INCLUDE ADLLOAD(DAZDEBUG)
INCLUDE ADLLOAD(DAZZAP)
NAME DAZNUCU(R)
```

You may use the JCL in the ADL Source Library member ADLINS5U as an example.

Step 6

This step need only be performed where one or more of the application programs to be converted uses the HLPI.

Create the executable ADL precompiler module by running the link editor and specifying the following link edit directives:

```
ORDER DAZPARM
INCLUDE ADLLOAD(DAZPARM)
INCLUDE ADLLOAD(DAZEXEC)
INCLUDE ADLLOAD(DAZSERV)
INCLUDE ADLLOAD(DAZLANP)
INCLUDE ADLLOAD(DAZSYXTB)
INCLUDE ADLLOAD(DAZDEBUG)
INCLUDE ADLLOAD(DAZZAP)
NAME DAZNUCP(R)
```

You may use the JCL in the ADL Source Library member `ADLINS6P` as an example.

Step 7

Create the executable `ADL CALLDLI` Interface batch module by running the link editor and specifying the following link edit directives. This step must also be performed when installing under IMS/TP:

```
ORDER DAZPARM
INCLUDE ADLLOAD(DAZPARM)
INCLUDE ADLLOAD(DAZBENT)
INCLUDE ADLLOAD(DAZAXES)
INCLUDE ADLLOAD(DAZSERV)
INCLUDE ADLLOAD(DAZINICB)
INCLUDE ADLLOAD(DAZCDPOS)
INCLUDE ADLLOAD(DAZDREIN)
INCLUDE ADLLOAD(DAZDEBUG)
INCLUDE ADLLOAD(DAZZAP)
NAME DAZNUCB(R)
```

You may use the JCL in the ADL source library member `ADLINS7B` as an example.

Step 8

Create the executable ADL Consistency Interface module for batch by running the linkage-editor and specifying the following link-edit directives:

```
ORDER DAZPARM
INCLUDE ADLLOAD(DAZPARM)
INCLUDE ADLLOAD(DAZAXES)
INCLUDE ADLLOAD(DAZBDOKE)
INCLUDE ADLLOAD(DAZBRQH)
INCLUDE ADALOAD(ADAUSER)
INCLUDE ADLLOAD(DAZCDPOS)
INCLUDE ADLLOAD(DAZCONSI)
INCLUDE ADLLOAD(DAZDEBUG)
INCLUDE ADLLOAD(DAZDREIN)
INCLUDE ADLLOAD(DAZINICB)
INCLUDE ADLLOAD(DAZSERV)
INCLUDE ADLLOAD(DAZZAP)
NAME DAZNUCA(R)
```

You may use the JCL in the ADL source library member `ADLINS8A` as an example.

The `ADAUSER` object module must be included from a valid Adabas load library.

Note:
Whenever you want the user written user exit (`DAZUEX01`) to become active in batch you must, in addition, include your user exit `DAZUEX01` within the ADL nucleus `DAZNUCA`. This may be achieved by

specifying an extra statement for the linkage-editor input:

```
INCLUDE USRLOAD(DAZUEX01)
```

See the section *ADL User Exit DAZUEX01* later in this documentation for more details on the purpose and conventions for the user exit DAZUEX01.

Step 9

Create the executable ADL Interfaces CICS module by running the link editor and specifying the following link edit directives:

```
INCLUDE CICLOAD(DFHEAI)
INCLUDE CICLOAD(DFHEAI0)
INCLUDE ADLLOAD(DAZPARM)
INCLUDE ADLLOAD(DAZCAPRI)
INCLUDE ADLLOAD(DAZCIFP)
INCLUDE ADLLOAD(DAZCLUB)
INCLUDE ADLLOAD(DAZCRQH)
INCLUDE ADLLOAD(DAZCDOKE)
INCLUDE ADLLOAD(DAZCONSI)
INCLUDE ADLLOAD(DAZAXES)
INCLUDE ADLLOAD(DAZSERV)
INCLUDE ADLLOAD(DAZINICB)
INCLUDE ADLLOAD(DAZCDPOS)
INCLUDE ADLLOAD(DAZDREIN)
INCLUDE ADLLOAD(DAZDEBUG)
INCLUDE ADLLOAD(DAZSTUB)
INCLUDE ADLLOAD(DAZZAP)
ENTRY   DAZPARM
NAME DAZNUCC(R)
```

You may use the JCL in the ADL source library member ADLINS9C as an example.

Note:
The ADL load library contains modules which are CICS release dependent. The members in the standard load library are for CICS TS 3.1 and CICS TS 3.2. When you run under CICS TS 2.3 or below, the ADL.LC23 library must be concatenated in front of the ADL load library.

Note:
Whenever you want the user written user exit (DAZUEX01) to become active under CICS, you must, in addition, include your user exit DAZUEX01 within the ADL Interfaces CICS nucleus, DAZNUCC. This may be achieved by specifying an extra statement for the linkage-editor input:

```
INCLUDE USRLOAD(DAZUEX01)
```

See the section *ADL User Exit DAZUEX01* later in this documentation for more details on the purpose and conventions for the user exit DAZUEX01.

Step 10

Create the executable ADL batch region controller by running the linkage-editor and specifying the following link-edit directives:

```
ENTRY DAZBIFP
INCLUDE ADLLOAD(DAZBIFP)
INCLUDE ADLLOAD(DAZBRQH)
INCLUDE ADALOAD(ADAUSER)
INCLUDE USRLOAD(DAZUEX01)
NAME DAZIFP(R)
```

You may use the JCL in the ADL source library member `ADLINS10` as an example. The ADL load library, the Adabas load library and the user load library must be defined with the file names `ADLLOAD`, `ADALOAD` and `USRLOAD`, respectively.

Note:
This step has to be performed only if you want a user written user exit (`DAZUEX01`) to become active in batch. See the section *ADL User Exit DAZUEX01* later in this documentation for more details on the purpose and conventions for this user exit.

Creating the Consistency Front-Ends (Steps 11 - 12)

Step 11

Note:
This step is only required if you plan to use the ADL Consistency Interface in batch.

Assemble and link-edit the ADL substitute for the Adabas link module in batch, `ADALNK`. You may use the JCL in the ADL source library member `ADLINS11` as an example. You need to perform this step only if you plan to use the ADL Consistency Interface. See the section *Batch Installation and Operation* in the *ADL Interfaces* documentation for more information on how to install the ADL Consistency Interface.

The ADL source library member to be assembled is `DAZLNKO`. Before the assembly, you will have to customize the assembler local variable `&ADALNK` in the ADL source library member `DAZLNKO`. `&ADALNK` specifies the new name chosen for the original Adabas link module which must be renamed. It is defaulted to `"ADAOLK"`.

Note that the ADL source library, the Adabas source library and the system macro libraries must be concatenated in the given order.

If you want to make use of the table of converted Adabas files (`DAZTCF`) you have to perform the steps described in section *Batch Installation and Operation* in the *ADL Interfaces* documentation. Assemble `DAZTCF` before link-editing `ADALNK`. The assembly of `DAZTCF` is included in the sample `JCL` `ADLINS11`. Add the following statement to the linkage-editor input for the Adabas batch link module:

```
INCLUDE ADLLOAD(DAZTCF)
```

For Adabas version 8 and above, you must additionally link the Adabas link-globals-table LNKGBLS to the ADL substitute:

```
INCLUDE ADALOAD(LNKGBLS)
```

Step 12

This step is only required if you plan to use the ADL Consistency Interface under CICS. See the section *CICS Installation and Operation* in the *ADL Interfaces* documentation for more information on how to install the ADL Consistency Interface.

Add the following statement to the linkage-editor input for the Adabas link module under CICS or, for
Adabas version 8, to the linkage-editor input for the Adabas globals module:

```
INCLUDE ADLLOAD(ADLEXITB)
```

If you want to use the table of converted Adabas files, assemble the `DAZTCF` table and include this also:

```
INCLUDE ADLLOAD(DAZTCF)
```

Note:
The parameter `LUSAVE` in the Adabas link source member `LNKOLSC` must have a value of at least 72
bytes. For Adabas Version 8 and above set the "ADL" parameter in the LGBLSET macro to "YES".