z/VSE 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 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/VSE 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 JCS given as an example below.

Library Space Requirements (z/VSE)

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

Library	3390
SL	40, 1
RL	80, 1
CL	80, 1

Copying the Tape Contents to a z/VSE 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 with your local naming conventions.

The JCL in this member 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 member COPYTAPE.JOB contains the JCL to unload the datasets for all delivered products from the tape to your disk, except the datasets that you can directly install from tape, for example, Natural INPL objects.

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

- Step 1 - Copy Dataset COPYTAPE.JOB from Tape to Disk
- Step 2 - Modify COPYTAPE.JOB
- Step 3 - Submit COPYTAPE.JOB

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

The dataset COPYTAPE.JOB (File 5) contains the JCL to unload all other existing datasets from tape to disk. To unload COPYTAPE.JOB, use the following sample JCL:

```
* $$ JOB JNM=LIBRCAT,CLASS=0,                                          +
* $$ DISP=D,LDEST=(*,UID),SYSID=1
* $$ LST CLASS=A,DISP=D
// JOB LIBRCAT
```

```
* ************************************
*     CATALOG COPYTAPE.JOB TO LIBRARY
* ************************************
// ASSGN SYS004,NNN                              <------  tape address
// MTC REW,SYS004
// MTC FSF,SYS004,4
ASSGN SYSIPT,SYS004
// TLBL IJSYSIN,'COPYTAPE.JOB'
// EXEC LIBR,PARM='MSHP; ACC S=lib.sublib'       <------- for catalog
/*
// MTC REW,SYS004
ASSGN SYSIPT,FEC
/*
/&amp;
* $$ EOJ
```

Where:

NNN is the tape address
lib.sublib is the library and sublibrary of the catalog

Step 2 - Modify COPYTAPE.JOB

Modify COPYTAPE.JOB to conform with your local naming conventions and set the disk space parameters before submitting this job.

Step 3 - Submit COPYTAPE.JOB

Submit COPYTAPE.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.J 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.J 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.J 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 section *The ADL Parameter Module* .

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

Creating the ADL Executable Modules (Steps 5 - 10)

Step 5

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

```
PHASE DAZNUCU,*,NOAUTO
INCLUDE DAZPARM
INCLUDE DAZCCGEN
INCLUDE DAZCCOUT
INCLUDE DAZCCSUB
INCLUDE DAZAXES
INCLUDE DAZSERV
INCLUDE DAZDEBUG
INCLUDE DAZZAP
ENTRY DAZPARM
```

You may use the JCS in the ADL Source Library member ADLINS5U.J 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:

```
PHASE DAZNUCP,*,NOAUTO
INCLUDE DAZPARM
INCLUDE DAZEXEC
INCLUDE DAZSERV
INCLUDE DAZLANP
INCLUDE DAZSYXTB
INCLUDE DAZDEBUG
INCLUDE DAZZAP
ENTRY DAZPARM
```

You may use the JCS in the ADL Source Library member ADLINS6P.J 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:

```
PHASE DAZNUCB,*,NOAUTO
INCLUDE DAZPARM
INCLUDE DAZBENT
INCLUDE DAZAXES
INCLUDE DAZSERV
INCLUDE DAZINICB
INCLUDE DAZCDPOS
INCLUDE DAZDREIN
INCLUDE DAZDEBUG
INCLUDE DAZZAP
ENTRY DAZPARM
```

You may use the JCS in the ADL Source Library member ADLINS7B.J 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:

```
PHASE DAZNUCA,*,NOAUTO
INCLUDE DAZPARM
INCLUDE DAZAXES
INCLUDE DAZBDOKE
INCLUDE DAZBRQH
INCLUDE ADAUSER
INCLUDE DAZCDPOS
INCLUDE DAZCONSI
INCLUDE DAZDEBUG
INCLUDE DAZDREIN
INCLUDE DAZINICB
INCLUDE DAZSERV
INCLUDE DAZZAP
ENTRY DAZPARM
```

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

Note:
When 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 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.

You may use the JCS in the ADL source library member ADLINS8A.J as an example on how to link-edit the ADL nucleus DAZNUCA.

Step 9

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

```
PHASE DAZNUCC,*,NOAUTO
INCLUDE DFHEAI
INCLUDE DFHEAI0
INCLUDE DAZPARM
```

```
INCLUDE DAZCAPRI
INCLUDE DAZCIFP
INCLUDE DAZCRQH
INCLUDE DAZCDOKE
INCLUDE DAZCLUB
INCLUDE DAZCONSI
INCLUDE DAZAXES
INCLUDE DAZSERV
INCLUDE DAZINICB
INCLUDE DAZCDPOS
INCLUDE DAZDREIN
INCLUDE DAZDEBUG
INCLUDE DAZSTUB
INCLUDE DAZZAP
ENTRY DAZPARM
```

You may use the JCS in the ADL Source Library member `ADLINS9C.J` as an example.

Note:
When 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 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:

```
ACTION CLEAR
PHASE DAZIFP,*,NOAUTO
INCLUDE DAZBIFP
INCLUDE DAZBRQH
INCLUDE ADAUSER
INCLUDE DAZUEX01
ENTRY DAZBIFP
```

You may use the JCS in the ADL source library member `ADLINS10.J` as an example. The ADL load library, the Adabas load library and the user load library must be defined in a `"LIBDEF PHASE,SEARCH="` statement in the given order.

It is only necessary to perform this step, 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

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 JCS in the ADL source library member ADLINS11.J 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 DAZLNKD. Before the assembly, you will have to customize the assembler local variable &ADALNK in the ADL source library member DAZLNKD. &ADALNK specifies the new name chosen for the original Adabas link module. 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 of 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 DAZTCF
```

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

```
INCLUDE 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:

```
INCLUDE ADLEXITB
```

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

```
INCLUDE DAZTCF
```

Note:
The parameter LUSAVE in the Adabas link source member LNKOLSC must have a value of at least 72 bytes.