Adabas

Installation for z/VSE

Version 8.1.4

June 2014

Adabas

This document applies to Adabas Version 8.1.4.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: ADAMF-VSE-INSTALL-814-20140626

Table of Contents

1 Installation for z/VSE

This document is intended for those who plan or perform Adabas z/VSE installation, and for those who manage or maintain an Adabas database system (such as database administrators and systems programming personnel).

Notation `vrs`, `vr`, or `v`: When used in this documentation, the notation `vrs` or `vr` stands for the relevant version of a product. For further information on product versions, see *version* in the *Glossary*.

2 Supported Environments

For information on the support platforms for this release of Adabas, read *Supported Platforms*, in the *Adabas Release Notes*.

3 Installation Procedure

This section describes the procedure for Adabas installation in z/VSE environments.

Installation Checklist

The following is an overview of the steps for installing Adabas on a z/VSE system.

Step	Description	Additional Information
1	Allocate DASD space for the Adabas libraries.	The libraries are restored from the installation tape. Refer to the section *Disk Space Requirements for Libraries*.
2	Allocate DASD space for the Adabas database.	For better performance, distribute the database files over multiple devices and channels. Refer to the section *Disk Space Requirements for the Database*.
3	Specify a z/VSE partition for running the Adabas nucleus.	Refer to the section *Adabas Nucleus Partition/Address Space Requirements*.
4	Define the library before restoration.	See section *Defining the Library*.
5	Restore the Adabas libraries.	See section *Installing the Adabas Release Tape*.
6	Install the Adabas SVC using the ADASIP program.	See section *Initializing the Adabas Communication Environment*.
7	Create the sample JCS job control for installing Adabas.	See section *Prepare the Installation Sample JCS for Editing*
8	Customize and run job ADAIOOAL to link the Adabas options table for installation customization.	See section *Modify, Assemble, and Link the Adabas Options Table*
9	Customize and catalog the two procedures ADAV*v*LIB and ADAV*v*FIL before placing them back in the procedure library. The following specific items must be customized: ■ file IDs for the database and libraries; ■ volumes for libraries and database files; ■ space allocation for database files	See section *Catalog Procedures for Defining Libraries and the Database*
10	Customize and run ADAFRM to allocate and format the Adabas database.	Steps 10-19 require changes to the setup definitions as described in section *Database Installation Steps*
11	Customize and run ADADEF to define global database characteristics.	
12	Customize and run ADALODE, ADALODV, and ADALODM to load the demo files.	
13	Install the product license file.	

Step	Description	Additional Information
14	Customize and run ADANUC to start the Adabas nucleus to test Adabas communications.	
15	Customize and run ADAREP in MULTI mode with the `CPLIST` parameter to test Adabas partition communication.	
16	Customize and run ADAINPL to load the Adabas Online System, if used.	
17	Terminate the Adabas nucleus.	
18	Customize and run ADASAV to back up the database.	
19	Customize and run DEFAULTS to insert the ADARUN defaults with the ZAP utility.	
20	Install the required TP link routines for Adabas	See section ***Installing Adabas With TP Monitors***.

Contents of the Release Tape

The following table describes most of the libraries included on the release tape. Once you have unloaded the libraries from the tape, you can change these names as required by your site, but the following lists the names that are delivered when you purchase Adabas for z/VSE environments.

Library Name	Description
ADA*vrs*.EMPL	The Employees demo file, containing dummy employee data you can use for testing Adabas. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.ERRN	Error messages for the Adabas Triggers and Stored Procedures Facility. These messages can be viewed using the Natural SYSERR utility. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.INPL	The code for Adabas Online System, Adabas Caching Facility, Triggers and Stored Procedures Facility, and various add-on demo products. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.LC*nn*	The Adabas library containing character encoding members to support various languages and Unicode. The *nn* letters in the library name represents a number from "00" to "99", assigned by Software AG. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.LIBR	The library for Adabas. The *vrs* in the library name represent the *version* of Adabas.
ADA*vrs*.MISC	The Miscellaneous demo file, containing dummy miscellaneous data you can use for testing Adabas. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.PERL	The LOB demo file storing the LOB data referenced by the Personnel demo file. The *vrs* in the library name represent the *version* of Adabas.
ADA*vrs*.PERS	The Personnel demo file, containing dummy personnel data you can use for testing Adabas. This demo file includes fields that make use of the extended and expanded

Library Name	Description
	features of Adabas 8, include large object (LOB) fields. The *vrs* in the library name represents the *version* of Adabas. **Note:** The Personnel demo file must be installed on a UES-enabled database because it includes wide-character format (W) fields.
ADA*vrs*.VEHI	The Vehicles demo file, containing dummy vehicle data you can use for testing Adabas. The *vrs* in the library name represents the *version* of Adabas.
APS*vrs*.L017	A Software AG internal library. The *vrs* in the library name represents the *version* of the internal library code, which is not necessarily the same as the version of Adabas.
APS*vrs*.LIBR	A Software AG internal library. The *vrs* in the library name represents the *version* of the internal library code, which is not necessarily the same as the version of Adabas.
MLC*vrs*.LIBJ	The sample job library for Software AG's common mainframe license check software. The *vrs* in the library name represents the *version* of the license check software, which is not necessarily the same as the version of Adabas.
MLC*vrs*.LIBR	The load library for Software AG's common mainframe license check software. The *vrs* in the library name represents the *version* of the license check software, which is not necessarily the same as the version of Adabas.
WAL*vrs*.LIBR	The library for Adabas components shared by Adabas and other Software AG products, such as Entire Net-Work. The *vrs* in the library name represents the *version* of Adabas.
WCA*vrs*.DOSLIBR	The library for Entire Net-Work Administration, used by some of the Adabas add-on products. The *vrs* in the library name represents the *version* of Entire Net-Work Administration.

Adabas is shipped with the code for Entire Net-Work Client (open systems software) and Entire Net-Work Administration (mainframe software). Entire Net-Work Client and Entire Net-Work Administration are Software AG middleware packages used for communication between Adabas or Event Replicator Server databases on the mainframe and open systems software packages such as Adabas Manager (including the Adabas Manager demo) or Event Replicator Administration. Entire Net-Work Administration is a limited version of Entire Net-Work for mainframes and includes the Simple Connection Line Driver.

Note: Entire Net-Work Client requires a license key. A limited license is shipped with your Adabas software to support the Adabas Manager demo. If you purchase a full version of Adabas Manager, you will need a full license of Entire Net-Work Client.

If appropriate Entire Net-Work mainframe and client products are not already installed on your system, install Entire Net-Work Administration on the mainframe and Entire Net-Work Clienton the client side. For complete information on these products, read the Entire Net-Work Administration documentation and Entire Net-Work Client Administration.

Preparing to Install Adabas

The major steps in preparing for Adabas installation are

- checking for the correct prerequisite system configuration; and
- allocating disk and storage space.

The following sections describe the nominal disk and storage space requirements, and how to allocate the space.

Disk Space Requirements for Libraries

The Adabas library requires a minimum of 3380/3390 disk space as shown below. A certain amount of extra free space has been added to the requirements for library maintenance purposes.

Library	3380 Tracks	3390 Tracks
Adabas Library	600	540

This space is needed for Adabas objects and phases as well as source and JCS samples.

Disk Space Requirements for the Database

The Adabas database size is based on user requirements. For more information, refer to *Adabas DBA Tasks*. Suggested sizes for an initial Adabas database, allowing for limited loading of user files and the installation of Natural, are as follows.

The minimum 3380 disk space requirements are:

Database Component	3380 Cylinders Required	3380 Tracks Required
ASSOR1 (Associator)	30	450
DATAR1 (Data Storage)	70	1050
WORKR1 (Work space)	10	150
TEMPR1 (temporary work space)	15	225
SORTR1 (sort work space)	15	225

The minimum 3390 disk space requirements are:

Database Component	3390 Cylinders Required	3390 Tracks Required
ASSOR1 (Associator)	26	390
DATAR1 (Data Storage)	62	930
WORKR1 (Work space)	10	150
TEMPR1 (temporary work space)	14	210
SORTR1 (sort work space)	14	210

Data Sets Required for UES Support

The Software AG internal product libraries (APS - porting platform) are required if you intend to enable a database for universal encoding service (UES) support. These libraries are now delivered separately from the product libraries.

For UES support, the following libraries must be loaded and included in the LIBDEF concatenation:

```
APSvrs.LIBR
APSvrs.LOnn
```

where *vrs* is the *version* of the library provided on the most recent tape for these components and *aa* is LD, LC, or LS and *nn* is the load library level. If the library with a higher level number is not a full replacement for the lower level load library(s), the library with the higher level must precede those with lower numbers in the LIBDEF concatenation.

Also for UES support, the following library must be loaded and included in the session execution JCL:

```
ADAvrs.ADAvrsCS
```

For information about setting up connections to UES-enabled databases, see section ***Enabling Universal Encoding Support (UES) for Your Adabas Nucleus***, elsewhere in this guide.

Disk Space Requirements for Internal Product Data Sets

The minimum disk space requirements on a 3390 disk for the internal product libraries delivered with Adabas Version 8 are as follows:

Library	3390 Cylinders	3390 Tracks
ADA*vrs*.ADA*vrs*CS	30	450
APS*vrs*.LIBR	8	120
APS*vrs*.L0*nn*	5	75

Adabas Nucleus Partition/Address Space Requirements

The Adabas nucleus requires at least 900-1024 KB to operate. The size of the nucleus partition may need to be larger, depending on the ADARUN parameter settings. Parameter settings are determined by the user.

Defining the Library

It is necessary to define the library before restoration. The following two examples show how VSAM and non-VSAM libraries are defined.

Defining a VSAM Library

The following is a job for defining a VSAM library:

```
// JOB DEFINE DEFINE VSAM V8 ADABAS LIBRARY
// OPTION LOG
// EXEC IDCAMS,SIZE=AUTO
DEFINE CLUSTER -
(NAME(ADABAS.ADAvrs.LIBRARY) -
VOLUME(vvvvvv vvvvvv) -
NONINDEXED -
RECORDFORMAT(NOCIFORMAT) -
SHR(2) -
TRK(nnnnnn)) -
DATA (NAME(ADABAS.ADAvrs.LIBRARY.DATA))
/*
// OPTION STDLABEL=ADD
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY',,VSAM
// EXEC IESVCLUP,SIZE=AUTO
ADABAS.ADAvrs.LIBRARY
/*
// EXEC LIBR
DEFINE L=SAGLIB R=Y
DEFINE S=SAGLIB.ADAvrs REUSE=AUTO R=Y
LD L=SAGLIB OUTPUT=STATUS
```

```
/*
/&

—where
vvvvvv vvvvvv are the volumes for primary and secondary space.
nnnnnn is the number of tracks for primary and secondary space.
vrs is the Adabas version.
```

Notes:

1. For FBA devices the tracks (TRK...) operand is replaced by the blocks (BLOCKS...) operand.
2. SAGLIB is the name of the Adabas library. The name SAGLIB can be changed to suit user requirements.

Defining a Non-VSAM Library

The following is a job for defining a non-VSAM library:

```
// JOB DEFINE DEFINE NON-VSAM V8 ADABAS LIBRARY
// OPTION LOG
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY',2099/365,SD
// EXTENT SYS010,vvvvvv,1,0,ssss,nnnn
// ASSGN SYS010,DISK,VOL=vvvvvv,SHR
// EXEC LIBR
DEFINE L=SAGLIB R=Y
DEFINE S=SAGLIB.ADAvrs REUSE=AUTO R=Y
LD L=SAGLIB OUTPUT=STATUS
/*
/&

where:
SYS010 is the logical unit for Adabas library.
vvvvvv is the volume for Adabas library.
ssss is the starting track or block for specified library.
nnnn is the number of tracks or blocks for specified library.
vrs is the Adabas version.
```

Restoring the ADAvrs LIBR File

Restore the ADA*vrs* LIBR file into sublibrary SAGLIB.ADA*vrs*. See the next section for information about preparing modules to run without the ESA option active.

Note: See the *Report of Tape Creation* that accompanies the tape to position the tape to the correct file.

If you have a license for one of the following Software AG products, restore the file into the appropriate sublibrary:

Product	File	Sublibrary
Adabas Caching Facility (ACF)	ACF*vrs*.LIBR	SAGLIB.ACF*vrs*
Adabas Online System (AOS)	AOS*vrs*.LIBR	SAGLIB.AOS*vrs*
Adabas Parallel Services (ASM)	ASM*vrs*.LIBR	SAGLIB.ASM*vrs*
Adabas Delta Save Facility (ADE)	ADE*vrs*.LIBR	SAGLIB.ADE*vrs*

For information about installing these products, see the documentation for that product.

Using the ADAvrs LIBR File

Where applicable, modules for Adabas are shipped with `AMODE=31` active.

Storage Above or Below the 16-MB Limit

Adabas can acquire storage above the 16-megabyte addressing limit. This capability allows Adabas to acquire the buffer pool (`LBP`), work pool (`LWP`), format pool (`LFP`), and attached buffers (*NAB*) above 16 MB.

Where applicable, modules for Adabas are shipped with `AMODE=31` active. If you prefer to have buffers placed below the 16-megabyte limit, ADARUN must be relinked with `AMODE=24`.

User Program Execution in AMODE=31 and RMODE=ANY

Programs that will execute `AMODE=31` or `RMODE=ANY` must be relinked with the new ADAUSER object module.

In addition, because the IBM VSE LOAD macro cannot be issued in `RMODE=ANY`, the IBM VSE CDLOAD macro must be used. Therefore, the zap to change the ADAUSER CDLOAD to the LOAD macro cannot be used.

Installing Adabas in a z/VSE VM Guest System

Adabas may be installed and executed on a z/VSE system that runs as a guest under VM.

When running Adabas in this environment, the CPUID of the guest z/VSE system must not contain a value of X'FFFFFFFF'. It it does, the Adabas nucleus terminates abnormally during command queue processing.

Installing the Release Tape

This section explains how to:

- copy data set COPY.JOB from tape to disk
- modify this member according to your local naming conventions

The JCL in this member is then used to copy all data sets from the tape to disk.

If the data sets for more than one product are delivered on the tape, the member COPYTAPE.JOB contains JCL to unload the data sets for all delivered products from the tape to your disk, except the data sets that you can directly install from tape, for example, Natrural INPL objects.

You can use the modified data set to copy all data sets from tape to disk. You will then need to perform the individual install procedure for each component.

Note: If you are using SMA, please refer to Installing Software AG Products with SMA in the *System Maintenance Aid* documentation. If you are not using SMA, please follow the instructions below.

Step 1: Copy Data Set COPYTAPE.JOB From Tape To Disk

The data set COPYTAPE.JOB (file 5) contains the JCL to unload all other existing data sets 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
```

```
/*
/&
* $$ EOJ

--- where
 nnn is the tape address, and 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 data sets from the tape to your disk.

Initializing the Adabas Communication Environment

Communication between the Adabas nucleus residing in a z/VSE partition and the user (either a batch job or TP monitor such as Com-plete or CICS) in another partition is handled with an Adabas SVC (supervisor call).

The program ADASIP is used to install the Adabas SVC. The system can run ADASIP to dynamically install the SVC without an IPL. Special instructions apply when using z/VSE with the Turbo Dispatcher as described in the next section below.

For information about messages or codes that occur during the installation, refer to the *Adabas Messages and Codes* documentation.

- ADASIP Runtime Display

Installing the Adabas SVC with Turbo Dispatcher Support

The Adabas SVC module supports the IBM z/VSE Turbo Dispatcher environment.

In a Turbo Dispatcher environment, the Adabas SVC runs in parallel mode when entered. Adabas processes multiple SVC calls made by users in parallel.

ADASIP Processing

To enable Turbo support, ADASIP installs a z/VSE first-level interrupt handler (ADASTUB) that screens all SVCs. When ADASTUB finds an Adabas SVC, it passes control directly to the Adabas SVC.

If your system is capable of running the Turbo Dispatcher and you do not want to run a particular SVC through the Turbo interface, you can set the UPSI flag V to 1 to exclude a particular SVC from use through the Turbo interface. See the ADASIP UPSI statement.

You can activate the ADABAS SVC with multiple CPUs active by specifying UPSI C. ADASIP will dynamically de-activate and re-activate the CPUs if required. If multiple CPUs are active and the UPSI C has not been specified, the following messages will be displayed:

```
ADASIP60 Only 1 CPU can be active during ADASIP
ADASIP79 Should we stop the CPUs? (yes/no)
```

Answering yes to this message will allow activation to occur; the CPUs will be dynamically de-activated and re-activated. Answering no will terminate ADASIP.

The ADASTUB module is installed only once per IPL process. On the first run of a successful ADASIP, the following set of messages are returned:

```
ADASIP63 ADASTUB Module Loaded at nnnnnnnn
ADASIP78 VSE Turbo Dispatcher Version nn
ADASIP69 Turbo Dispatcher Stub A C T I V E
```

When running ADASIP for subsequent Adabas SVC installations, the following message is displayed for information only:

```
ADASIP74 Info : Stub activated by previous ADASIP
```

When dynamically re-installing an Adabas SVC that was previously installed with Turbo Dispatcher support, execute a SET SDL for the Adabas SVC only. Do not execute the SET SDL for ADANCHOR a second time.

Note: Repeated re-installations of an Adabas SVC without an IPL may result in a shortage of 24-bit GETVIS in the SVA.

Running ADASIP

ADASIP requires a prior SET SDL for the SVC, and therefore must run in the BG partition. To install the Adabas SVC without an IPL, execute the following JCS in BG.

Notes:

1. When using the EPAT Tape Management System, EPAT must be initialized before running ADASIP.
2. At execution time, the ADASIP program determines if a printer is assigned to system logical unit SYSLST. If no printer is assigned, messages are written to SYSLOG instead of SYSLST.

For information about the ADASIP parameters, see the section ***ADASIP Execution***.

To automatically install the Adabas SVC during each IPL, insert the following JCS (or its equivalent) into the ASI BG JCS procedure immediately before the START of the POWER partition where

nn	is the number of IDT entries
suffix	is the optional two-byte suffix for the z/VSE SVC name to be loaded by ADASIP. The previous z/VSE SVC version must be linked with a different suffix.
svc	is an available SVC number in your z/VSE system to be used as the Adabas SVC.
volume	is the specified volume for the Adabas library.
vrs	is the Adabas *version*

Without Turbo Dispatcher Support

The following sample is available in member ADASIP.X:

```
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY'
// EXTENT SYS010,volume
// ASSGN SYS010,DISK,VOL=volume,SHR
// LIBDEF PHASE,SEARCH=SAGLIB.ADAvrs
SET SDL
ADASVCvr,SVA
/*
// OPTION SYSPARM='svc,suffix' SVC NUMBER
```

```
// UPSI 00000000 UPSI OPTIONS FOR ADASIP
// EXEC ADASIP,PARM='NRIDTES=nn'
```

With Turbo Dispatcher Support

The following sample is available in member ADASIPT.X:

```
// JOB ADASIPT INSTALL THE ADABAS SVC (TURBO)
// OPTION LOG,NOSYSDUMP
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY'
// EXTENT SYS010,volume
// ASSGN SYS010,DISK,VOL=volume,SHR
// LIBDEF PHASE,SEARCH=SAGLIB.ADAvrs
SET SDL
ADASVCvr,SVA
ADANCHOR,SVA
/*
// OPTION SYSPARM='svc,suffix' SVC NUMBER
// SETPFIX LIMIT=100K REQUIRED; SEE NOTE 2
// UPSI 00000000 UPSI OPTIONS FOR ADASIP
// EXEC ADASIP,PARM='NRIDTES=nn'
```

Notes:

1. A `SETPFIX` parameter is required with Turbo Dispatcher support to page fix ADASIP at certain points in its processing. A value of 100K should be adequate.
2. The SET SDL statement for ADANCHOR is required for Turbo Dispatcher support. This is in addition to the SET SDL statement for ADASVC*vr*.

Finding an Unused SVC

Adabas requires an entry in the z/VSE SVC table. To find an unused SVC, use one of the following methods:

Method 1

Set the S flag specified in the UPSI for ADASIP to create a list of used and unused SVCs in the z/VSE SVC table.

Method 2

Obtain a listing of the supervisor being used.

Using the assembler cross-reference, locate the label SVCTAB; this is the beginning of the z/VSE SVC table. The table contains a four-byte entry for each SVC between 0 and 150 (depending on the z/VSE version).

Locate an entry between 31 and 150 having a value of ERR21. This value indicates an unused SVC table entry. Use the entry number as input to ADASIP.

Loading a Secondary Adabas SVC

You can optionally specify a suffix to indicate the version of an SVC, as shown in the previous JCS examples. This allows you to run two different versions of the SVC. Before specifying a suffix, however, you must have previously linked the second version of the SVC. In addition, you must have performed a SET SDL operation on the new SVC's name (for example, ADASVC*xx*).

To optionally specify a different Adabas SVC using ADASIP, specify the SVC suffix (the last two bytes in the form, ADASVC*xx*), as follows, where *xx* is the two-byte suffix of the new SVC:

```
// OPTION SYSPARM='svc,xx'
```

ADASIP Execution Parameters

This section describes the ADASIP execution parameters.

- OPTION SYSPARM= Statement
- UPSI Statement
- NRIDTES PARM= Option
- REPLACE PARM= Option
- DMPDBID PARM= Option

Runtime Display

OPTION SYSPARM= Statement

An optional correction (zap) can be applied to the Adabas ADASIP program to insert the default SVC so that no SYSPARM need be specified. See the section *Applying Zaps*.

SVC	The Adabas SVC number chosen must be unused by z/VSE or any other third party products (see the section *Finding an Unused SVC*).
SUFFIX	An optional two-byte value used to load a new version of the Adabas z/VSE SVC (see the section *Loading a Secondary Adabas SVC).*.

UPSI Statement

```
// UPSI DSxTVOGx
```

Setting the UPSI byte is the user's responsibility. If the UPSI byte is not set, the SVC installation executes normally.

The UPSI byte is used to select the following options:

Option	If option is set to 1
D	ADASIP dumps the Adabas SVC and ID table using PDUMP. This option should be used only after the SVC is installed.
S	ADASIP dumps the z/VSE SVC table and indicates whether each SVC is used or unused. No SVC number is required when using this function of ADASIP.
T	ADASIP dumps the z/VSE SVC table and the z/VSE SVC mode table.
V	The SVC is excluded from use through the Turbo interface.
O	Override the messages that ask if you wish to stop the processors when more than one processor is active. If you choose to override, the processors will be automatically stopped during ADASIP execution and restarted upon ADASIP termination.
G	ADASIP will display SYSTEM GETVIS allocation totals.

NRIDTES PARM= Option

The size of the ID table default supports up to 10 Adabas targets. However, the ADASIP program will allow you to increase this number by using this new option of the PARM operand on the EXEC card. To increase the size of the ID table to *nn* entries, specify the following when executing ADASIP:

```
// EXEC ADASIP,PARM='NRIDTES=nn'
```

where *nn* is the number of databases to be supported. Refer to the section ***Acquiring Storage for the ID Table*** for information about calculating the correct value for *nn*.

REPLACE PARM= Option

Specifying REPLACE=N or NO will cause warning messages ADASIP80 and ADASIP81 to appear if the SVC has been previously installed. Specifying REPLACE=Y or YES replaces the current SVC regardless of any active targets. The default value is REPLACE=NO. No abbreviation of the REPLACE keyword is supported.

Caution: Setting the REPLACE parameter to YES should be done carefully. Replacing an SVC while your targets are running can produce unpredictable results.

If both the NRIDTES and REPLACE keywords are specified, they must be separated by a comma. For example:

```
//EXEC ADASIP,PARM='NRIDTES=10,REPLACE=YES'
```

DMPDBID PARM= Option

This ADASIP option allows snap dumps of the Adabas command queue for a specified database ID (DBID). The dump is written to SYSLST. The OPTION SYSPARM statement must specify the SVC number to perform the snap dump. For example, to perform a snap dump of the database 5 command queue, issue:

```
// OPTION SYSPARM='svc,suffix'
// EXEC ADASIP,PARM='DMPDBID=5'
```

ADASIP Runtime Display

When ADASIP is run, the ADASIP00 message displays the current system level.

```
ADASIP00 ...ADABAS V8 VSE SIP STARTED
SIP IS RUNNING UNDER VSE/systype-mode
ADASIP00 ... (yyyy-mm-dd, SM=sm-level, ZAP=zap-level)
ADASIP00 ... SIP IS RUNNING UNDER OSYS LEVEL Vnnn
ADASIP00 ... SIP IS LOADING ADABAS SVC LEVEL Vnnn
ADASIP00 ... ADASIP IS LOADING ADABAS SVC AMODE=amode
```

Installing the Adabas Database

This section describes installation of the Adabas database for z/VSE systems. Note that all applicable early warnings and other fixes must first be applied. For descriptions of any messages or codes that occur, refer to the *Adabas Messages and Codes* documentation.

- Prepare the Installation Sample JCS for Editing
- Modify, Assemble, and Link the Adabas Options Table
- Catalog Procedures for Defining Libraries and the Database

- Database Installation Steps

Prepare the Installation Sample JCS for Editing

Note: This step is only necessary if the library cannot be edited directly.

The following sample installation job is available in member INSTALL.X.

Run the following job to load the installation samples:

```
* $$ JOB JNM=PUNINST,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
* $$ PUN CLASS=p,DISP=D
// JOB PUNINST INSTALL SAMPLES FOR ADABAS
// OPTION LOG
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volume,SHR
// EXEC LIBR
ACCESS SUBLIB=SAGLIB.ADAvrs
PUNCH ADAPROC.X /* PROCS FOR FILE AND LIBRARY DEFINITIONS */
PUNCH ADAIOOAL.X /* ADABAS OPTIONS TABLE CUSTOMIZATION */
PUNCH ADASIP.X /* ADASIP JOB (NON-TURBO DISPATCHER) */
PUNCH ADASIPT.X /* ADASIP JOB (TURBO DISPATCHER) */
PUNCH ADAFRM.X /* SAMPLE ADAFRM JOB */
PUNCH ADADEF.X /* SAMPLE ADADEF JOB */
PUNCH ADALODE.X /* LOAD DEMO FILE EMPLOYEES */
PUNCH ADALODV.X /* LOAD DEMO FILE VEHICLES */
PUNCH ADALODM.X /* LOAD DEMO FILE MISC */
PUNCH ADALODP.X /* LOAD DEMO FILES PERSONNEL & LOB */
PUNCH ADANUC.X /* SAMPLE NUCLEUS STARTUP */
PUNCH ADAREP.X /* SAMPLE ADAREP JOB */
PUNCH ADAINPL.X /* SAMPLE NATINPL TO INSTALL AOS */
/*
/&
* $$ EOJ
```

where *p* is the output class for punch, *volume* is the specified volume for the Adabas library, and *vrs* is the Adabas *version*.

Once the selected members in the INSTALL job are within the local editor facility, the customization can begin.

Modify, Assemble, and Link the Adabas Options Table

Customize and run job ADAIOOAL to assemble and link the Adabas options table for installation customization.

The following describes the IORDOSO macro, which must be assembled and linked to the Adabas sublibrary as PHASE ADAOPD. The member X.ADAIOOAL shipped with Adabas can be used for this purpose.

IORDOSO Macro Overview

The IORDOSO macro allows you to customize Adabas operation in the following areas:

- Loading phases;
- IDRC compaction support for 3480 and 3490 tape devices;
- Interfaces to z/VSE disk space managers such as DYNAM/D;
- Interfaces to z/VSE tape managers such as DYNAM/T
- An option controlling how the system writes to fixed block addressing (FBA) devices;
- An option to write printer (PRINT and DRUCK) files under either DTFPR or DTFDI control;
- GETVIS message printing;
- Optional job exit processing;
- Options for controlling the creation of z/VSE JCS with the Adabas Recovery Aid utility ADARAI;
- Sequential file processing under VSAM/SAM;
- Input device control with SYS000 assignment;
- Name of external sort program.

IORDOSO Macro Parameters

The following parameters can be set in using the IORDOSO macro.

CDLOAD

Parameter	Description
CDLOAD={ NO | YES } ↩	Determines whether Adabas uses the CDLOAD (SVC 65) or the LOAD SVC (SVC 4) to load modules.

COMPACT

Parameter	Description
COMPACT={ NO | YES } ↩	If a sequential protection log (SIBA) is assigned to a 3480 or 3490 tape device, COMPACT=YES writes the SIBA in IDRC compaction mode. The default is COMPACT=NO (no compaction).

DISKDEV

Parameter	Description
DISKDEV=*devtype* ↩	Specifies the device type on which space for sequential files is to be allocated (see notes 1 and 2).

Notes:

1. Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.
2. Valid disk device types are 3380, 3390, 9345 and FBA.

DISKMAN

Parameter	Description
DISKMAN={ NO | YES }	Indicates to Adabas that a z/VSE disk space manager such as DYNAM/D is active. If DISKMAN=YES is specified, DISKDEV or DISKSYS must also be specified.

DISKSYS

Parameter	Description
DISKSYS=*sysnum* ↩	When a disk space manager such as DYNAM/D is present, use the DISKSYS parameter to specify the programmer logical unit (LUB). The specified value, which can be from 000 to 255, determines the disk device type for the SAM or VSAM sequential file. There is no default value.

DISKTYP

Parameter	Description
DISKTYP=*text*	This parameter is for information only, and is processed as a comment. The value *text* can be up to 16 bytes long.

DTFDI

Parameter	Description
DTFDI={ NO | YES } ↩	DTFDI=YES directs the PRINT (SYSLST) and DRUCK (SYS009) output to be device-independent, causing all ADARUN, ADANUC, session, statistics, and utility output to be written to where SYSLST or SYS009 is assigned (printer, disk, or tape). When you specify DTFDI=YES, the PRTDSYS and PRTRSYS parameters are ignored. If you specify DTFDI=NO (the default), output is directed using DTFPR.

FBAVRF

Parameter	Description
FBAVRF={ NO | YES } ↩	FBA users only: the FBAVRF parameter specifies whether Adabas does WRITE VERIFY I/O commands, or normal WRITEs. If FBAVRF=YES is specified, WRITE VERIFY I/Os are performed; the default is normal WRITE operation.

GETMMSG

Parameter	Description
GETMMSG={ NO | YES } ↩	Determines whether or not z/VSE ADAIOR GETMAIN (GETVIS) messages are printed. No printing is the default.

JBXEMSG

Parameter	Description
JBXEMSG={ NO| YES | PRT }	The z/VSE parameter JBXEMSG determines whether job exit utility error messages are printed (JBXEMSG=PRT), displayed (JBXEMSG=YES, the default), or not presented (JBXEMSG=NO).

JBXIMSG

Parameter	Description
JBXIMSG={ NO | YES | PRT }	The z/VSE parameter JBXIMSG determines whether job exit utility information messages are printed (JBXIMSG=PRT, the default), displayed (JBXIMSG=YES), or not presented (JBXIMSG=NO).

JOBEXIT

Parameter	Description
JOBEXIT={ NO | YES }	JOBEXIT=YES activates the Adabas job exit utility, allowing any * SAGUSER job control statements to override the normal job input.

PFIXRIR

Parameter	Description
PFIXRIR={ NO | YES }	Specifies whether or not ADAMPM is page fixed in storage during the nucleus initialization process.

PRTDSYS

Parameter	Description
PRTDSYS={ *sysnum* ↩ | SYSLST } ↩	Specifies the programmer logical unit (LUB), and may be any number 000 - 254. If specified, the *sysnum* value replaces the default where the ADARUN messages are printed, which is SYSLST. The value specified by *sysnum* must be assigned in the partition before running the ADARUN program. For example:

Parameter	Description
	PRTDSYS=050 . // ASSGN SYS050,PRINTER

PRTRSYS

Parameter	Description
PRTRSYS={ *sysnum* | SYS009 } ↩	Specifies the programmer logical unit (LUB). If specified, this *sysnum* value replaces the default where the Adabas utility (DRUCK) messages are printed, which is SYS009.

RAIDASG

Parameter	Description
RAIDASG={ NO | YES } ↩	RAIDASG=YES specifies that the Adabas Recovery Aid (ADARAI) is to create z/VSE disk ASSGN statements. Such statements are sometimes not needed with a z/VSE disk manager facility.

RAITASG

Parameter	Description
RAITASG={ NO | YES }	RAITASG=YES specifies that the Adabas Recovery Aid (ADARAI) is to create z/VSE tape ASSGN statements. Such statements are sometimes not needed with a z/VSE tape manager facility.

SORTPGM

Parameter	Description
SORTPGM={ *sortpgm* | SORT } ↩	Specifies the name of the external sort program to be invoked during execution of the Adabas changed-data capture utility ADACDC. The default name is SORT.

SYS000O

Parameter	Description
SYS0000={ NO | YES } ↩	If SYS0000=NO (the default) is specified, the ADARUN statements are read normally. If SYS0000=YES is specified, Adabas determines the correct DTF for opening, depending on where SYS000 is assigned, as follows: Medium - SYS000 DTF Type Card DTFCD Disk DTFSD Tape DTFMT

TAPEDEV

Parameter	Description
TAPEDEV=*devtype* ↩	Specifies the tape device type on which sequential files are written (see notes 1 and 2).

Notes:

1. Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.
2. Valid tape device types are 2400, 3410, 3420, 3480 and 8809. For device types 3480, 3490, 3490E or 3590, specify TAPEDEV=3480.

TAPEMAN

Parameter	Description
TAPEMAN={ NO | YES }	Indicates that a z/VSE tape manager such as DYNAM/T is active. If TAPEMAN=YES is specified, TAPEDEV or TAPESYS must also be specified.

TAPESYS

Parameter	Description
TAPESYS=*sysnum*	When a tape manager such as DYNAM/T is present, this parameter is used to specify the programmer logical unit (LUB). The specified value, which can be any value from 000 to 255, determines the tape device type for the sequential file (see note). There is no default value.

Note: Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.

TAPETYP

Parameter	Description
TAPETYP=*text*	This parameter is for information only, and is processed as a comment. The value *text* can be up to 16 bytes long.

VSAMDEV

Parameter	Description
VSAMDEV=*devtype*	Specifies the disk device type on which VSAM/SAM space is to be allocated (see notes 1 and 2).

Notes:

1. Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.
2. Valid disk device types are 3380, 3390, 9345 and FBA.

VSAMSEQ

Parameter	Description
VSAMSEQ={ NO | YES }	Specifies whether sequential files are to be under the control of VSAM/SAM software. If VSAMSEQ=YES is specified, either VSAMDEV or VSAMSYS must also be specified.

VSAMSYS

Parameter	Description
VSAMSYS=*sysnum*	Specifies the programmer logical unit (LUB). The specified value, which can from 000 to 255, determines the device type for the sequential file written to VSAM/SAM space (see note). There is no default value.

Note: Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.

Additional Parameters Used for Internal Control Only

Three additional parameters are also available but are used only for internal control and should not be changed from their default settings unless otherwise specified by your Software AG technical support representative:

```
IORTRAC={NO | YES}
IORTSIZ={3000 | tablesize}
IORTTYP=(1... 14)(, opt1 ... opt14 ).
```

Catalog Procedures for Defining Libraries and the Database

Note: Sample JCS is available in ADAPROC.X

The job ADAPROC is divided into two procedures:

- ADAV*v*LIB defining the library or libraries; and
- ADAV*v*FIL defining the database.

Customize and catalog the two procedures before placing them back in the procedure library. The following specific items must be customized:

- file IDs for the database and libraries;

- volumes for libraries and database files;
- space allocation for database files.

The Adabas DEMO database files include ASSO, DATA, WORK, TEMP, SORT, CLOG, and PLOG.

Database Installation Steps

Follow the steps outlined below to install a new Adabas database under z/VSE.

- Step 1. Allocate and format the DEMO database.
- Step 2. Define the global database characteristics.
- Step 3. Load the demonstration (demo) files.
- Step 4. Install the product license file.
- Step 5. Start the Adabas nucleus and test the Adabas communications.
- Step 6. Test Adabas partition communications.
- Step 7. Load the Adabas Online System, if used.
- Step 8. Terminate the Adabas nucleus.
- Step 9. Back up the database.
- Step 10. Insert the ADARUN defaults.
- Step 11. Install the required TP link routines for Adabas.

Notes:

1. For information about running ADADEF, ADAFRM ADALOD, ADAREP, and ADASAV in steps 1-3, 5, and 8 below, see the *Adabas Utilities* documentation.
2. For information about customizing the nucleus job and about starting, monitoring, controlling, and terminating the nucleus, see the *Adabas Operations* documentation.

Step 1. Allocate and format the DEMO database.

Note: Sample JCS is available in ADAFRM.X

Customize and run the ADAFRM utility job to format the DEMO database areas. The following specific items must be customized:

- the Adabas SVC number, the database ID, and database device type(s);
- sizes of the data sets for each ADAFRM statement.

Step 2. Define the global database characteristics.

Note: Sample JCS is available in ADADEF.X

Customize and run the ADADEF utility job to define the global definition of the database. The following items must be customized:

- the Adabas SVC number, the database ID, and database device type(s);
- ADADEF parameters.

Step 3. Load the demonstration (demo) files.

Note: Sample JCS is available in ADALODE.X, ADALODV.X, ADALODM.X, and ADALODP.X.

Customize and run the job

- ADALODE to load the sample demo file EMPL;
- ADALODV to load the sample demo file VEHI;
- ADALODM to load the sample demo file MISC; and
- ADALODP to load the sample demo file PERS and its associated LOB demo file, PERL.

Note: The Personnel demo file must be installed on a UES-enabled database because it includes wide-character format (W) fields.

For each job, the following items must be customized:

- the Adabas SVC number, the database ID, and database device type(s);
- ADALOD parameters.

Step 4. Install the product license file.

The product license file is supplied on the individual customer installation tape or separately via an e-mail attachment. If the license file is provided on an installation tape, you can follow the instructions in this step to install the license file. If the license file is supplied via an e-mail attachment, you must first transfer the license to z/VSE, as described in *Transferring a License File from PC to a z/VSEHost Using FTP*, in *Adabas Operations Manual* and then you can install it, as described in this step.

Installing the license file.

In z/VSE environments, the product license file can be installed either as a load module or as a library member.

To install the product license file as a module, complete the following steps:

1 Verify that the license file is stored in an Adabas source library or sequential data set (with RECFM=F or FB and LRECL=80), taking care to preserve its format as ASCII.

2 If you loaded your Adabas license file to a library, review and modify sample JCS job ASMLICAL.X, adjusting the library and volume specifications as appropriate for your site. If you loaded your Adabas license file to a data set, use sample ASMLICAV.X instead.

Note: In sample jobs ASMLICAL.X and ASMLICAV.X, the standard label area is assumed to contain label information for library USERLIB. You can change this as appropriate for your library.

3 Submit modified sample job ASMLICAL.X or ASMLICAV.X.

These sample jobs generate your Adabas license in ADALIC.PHASE. They assume that ADALIC.PHASE will be in a user sublibrary. If a user sublibrary is chosen for ADALIC.PHASE, this sublibrary must be included in the LIBDEF search chain in your Adabas nucleus startup JCS. You may find it more convenient to place ADALIC.PHASE directly into the Adabas ADA*vrs* sublibrary, to avoid the need to define additional libraries. During initial testing, Software AG recommends using a user sublibrary.

To install the product license file as a library member, complete the following steps:

1 Verify that the license file is stored in an Adabas source library (with RECFM=F or FB and LRECL=80), taking care to preserve its format as ASCII.

2 Make sure any previously created ADALIC load module is inaccessible in the Adabas load library being used by the nucleus jobs. Adabas first tries to load ADALIC and if unsuccessful it reads from DDLIC.

3 Provide all Adabas nucleus startup jobs with a DLBL statement in the following format:

```
// DLBL DDLIC,'/libname/sublb/memname.memtype'
```

where *libname* is the Librarian name of the library, *sublib* is the name of the sublibrary, *memname* is the license member name, and *memtype* is the license member type.

To install the product license file as a sequential data set, complete the following steps:

1 Verify that the license file is stored in a sequential file (with RECFM=F or FB and LRECL=80), taking care to preserve its format as ASCII.

2 Make sure any previously created ADALIC load module is inaccessible in the Adabas load library being used by the nucleus jobs. Adabas first tries to load ADALIC and, if unsuccessful, it reads from DDLIC.

3 Provide all Adabas nucleus startup jobs with DLBL, EXTENT and ASSGN statements in the following format:

```
// DLBL DDLIC,'adabas.license.file'
// EXTENT SYSnnn
// ASSGN SYSnnn,DISK,VOL=volser,SHR
```

where *adabas.license.file* is the physical file name, *nnn* is an unused logical unit, and *volser* is the volume serial on which the license file resides.

Step 5. Start the Adabas nucleus and test the Adabas communications.

Note: Sample JCS is available in ADANUC.X.

Customize and run the job ADANUC to start up the Adabas nucleus. The following items must be customized:

- The Adabas SVC number, the database ID, and device type(s);

  

  Note: Be sure to include appropriate LIBDEF references for user sublibraries, especially the library containing the ADALIC license file. The licensing component MLC*vrs* must also be added to the LIBDEF SEARCH chain for load modules. These additional sublibraries can be added via the ADA*v*FIL procedure, as required.

- ADANUC parameters.

Step 6. Test Adabas partition communications.

Note: Sample JCS is available in ADAREP.X.

Customize and run the job ADAREP in MULTI mode with the `CPLIST` parameter to test Adabas partition communications. The following items must be customized:

- the Adabas SVC number, the database ID, and device type(s);
- ADAREP parameters.

Step 7. Load the Adabas Online System, if used.

Note: Sample JCS is available in ADAINPL.X. Read *Installing the AOS Demo Version*, elsewhere in this guide, and, if necessary, the installation section of the Adabas Online System documentation.

Customize and run the job ADAINPL to load the Adabas Online System into a Natural system file. A Natural file must first be created, requiring an INPL input file (see the Natural installation instructions). The following items must be customized:

- the Adabas SVC number, the database ID, and device type(s);
- the Natural INPL parameters and system file number.

Step 8. Terminate the Adabas nucleus.

Communicate with the Adabas nucleus (MSG F*n*) to terminate the session by entering the Adabas operator command `ADAEND` into the Adabas nucleus partition.

Step 9. Back up the database.

Customize and run the ADASAV utility job to back up the Version sample database. The following items must be customized:

- the Adabas SVC number, the database ID, and device type(s);
- ADASAV parameters.

Step 10. Insert the ADARUN defaults.

Optionally customize and run the DEFAULTS job to set the ADARUN defaults using the MSHP utility and to relink ADARUN. The following items may be customized:

- SVC number;
- database ID;
- device type(s).

Step 11. Install the required TP link routines for Adabas.

Refer to the section ***Installing Adabas With TP Monitors*** for the TP link routine procedure.

Migrating an Existing Database

Use the ADACNV utility to migrate existing databases to new releases of Adabas. See the *Adabas Utilities* documentation for more information.

Logical Unit Requirements

This section describes the Adabas logical unit requirements.

ADARUN

Logical Unit	File	Storage Medium
SYSLST	PRINT	Printer
SYS000	CARD	Tape / Disk
SYSRDR	CARD	Reader

Utility

Logical Unit	File	Storage Medium
SYS009	DRUCK	Printer
SYSIPT	KARTE	Reader

Nucleus

Logical Unit	File	Storage Medium
SYSLST	PRINT	Printer
SYSRDR	CARD	Reader

The highest logical unit used is SYS038 for the ADASAV utility. The programmer logical units default is described in the section ***Device and File Considerations***. The system programmer should review these requirements to ensure that there are enough programmer logical units to run the desired utilities in the desired partitions.

Job Exit Utility

Adabas provides a job exit to perform two different functions:

- Librarian input override processing

  The exit scans a job stream for Librarian input override statements. These statements indicate that card input (ADARUN CARD or utility KARTE statements) for a job step is to come from Librarian members rather than from SYSRDR or SYSIPT.

- ADARAI JCS capture processing

  The exit captures JCS before it is modified by tape or disk management systems for later use by ADARAI.

You can set the job exit to perform either function or both. By default, the job exit performs Librarian input override processing.

This section covers the following topics:

Installation and Initialization

The job exit can be installed during ASI processing or at any time afterward. It is installed in two steps:

to install the job exit:

1 Install programs SAGJBXT and SAGIPT in the SVA.

2 Run program SAGINST to initiate job exit processing.

You can include SAGJBXT in the $JOBEXIT list of eligible exits, but you must still place SAGIPT in the SVA and run SAGINST to allocate the required table(s).

SAGIPT runs above the 16-megabyte line if an appropriate 31-bit PSIZE is available. In addition, the table that stores information from input-override statements and/or the table that stores JCS for ADARAI use is placed in 31-bit GETVIS, if available.

SAGINST reads an input parameter that tells it whether to install the Librarian input override processing, ADARAI JCS capture processing, or both. The following parameter values are valid:

```
PARM=ADALIB (the default) installs Librarian input override processing
PARM=ADARAI installs ADARAI JCS capture processing
```

The following sample job control initializes the job exit:

Note: Sample JCS to initialize the job exit is available in member JBXTINST.X.

```
* $$ JOB JNM=SAGEXIT,CLASS=0
* $$ LST CLASS=A,DISP=D
// JOB SAGEXIT
// LIBDEF *,SEARCH=SAGLIB.ADAvrs
// EXEC PROC=ADAVvLIB
SET SDL
SAGJBXT,SVA
SAGIPT,SVA
/*
// EXEC SAGINST,PARM=ADARAI,ADALIB
/&
* $$ EOJ
```

where *vrs* is the Adabas *version*.

Librarian Input Override Processing

If Librarian input override processing is specified, the job exit scans a job stream for input override statements indicating that card input (ADARUN CARD or utility KARTE statements) for a job step is to come from Librarian members rather than from SYSRDR or SYSIPT. By default, the exit can store a maximum of 2000 input override cards simultaneously throughout the system. Adabas uses this facility when processing CARD and KARTE parameters.

Enable Librarian input override processing by adding * SAGUSER control statements to the job control stream between the // JOB and // EXEC statements.

A * SAGUSER statement can have three keyword parameters: `FILE`, `LIBRARY`, and `MEMBER`.

Keyword Syntax	Description
`FILE={ CARD | KARTE }`	The file to be read from a Librarian member. Specify "CARD" for ADARUN statements, or "KARTE" for utility statements.
`LIBRARY={ library.sublibrary ↩ | libdef.source }`	The library and sublibrary to be searched. If omitted, the current *libdef.source* chain is used.

Keyword Syntax	Description
MEMBER=*name* [,{ *type* | A }]	The member name and optionally the type to be read. If *type* is omitted, "A" is assumed.

The following is an example of a * SAGUSER control statement that specifies an alternate job exit member:

```
* SAGUSER FILE=CARD,MEMBER=NUC151
```

In the example above, Adabas searches the current *libdef.source* chain for member NUC151 with type A. If NUC151 is found, Adabas uses its contents as the nucleus startup parameters instead of SYSIPT.

To permit flexible startup processing, multiple SAGUSER statements may be specified for each file. In the following example, Adabas reads the input parameters first in member NUC151, then in member IGNDIB:

```
* SAGUSER FILE=CARD,MEMBER=NUC151
* SAGUSER FILE=CARD,MEMBER=IGNDIB
```

The following examples show the use of the LIBRARY parameter, and apply to z/VSE systems only:

```
* SAGUSER FILE=CARD,MEMBER=NUC151,LIBRARY=SAGULIB.TESTSRC
```

In the example above, Adabas searches sublibrary TESTSRC in the SAGULIB library for member NUC151 with type A. If NUC151 is not found in sublibrary TESTSRC of library SAGULIB, no further search is made. The DLBL and EXTENT information for the SAGULIB library must be available.

```
* SAGUSER FILE=CARD,MEMBER=NUC151.ADARUN,LIBRARY=SAGULIB.TESTSRC
```

In the example above, Adabas searches sublibrary TESTSRC in the SAGULIB library at nucleus initialization for member NUC151 with type ADARUN. The library member types PROC, OBJ, PHASE, and DUMP are not permitted.

Activating Adabas Use of Job Exit Processing

Specify JOBEXIT=YES to allow Adabas to use SAGUSER statements in the job stream and recatalog the Adabas options table (ADAOPD).

Using the Job Exit Utility for ADARAI JCS Capture

Once the job exit utility has been installed for ADARAI, all utilities that write information to the RLOG automatically obtain file information from the ADARAI table that the job exit maintains. Manual intervention is not required.

Job Exit Storage Requirements

The job exit requires from 84 to 298 kilobytes (KB) of SVA storage, depending on whether the Librarian input override interface and/or the ADARAI JCS interface is installed. Of that total,

- 2 kilobytes are used for program storage (PSIZE);
- 82-kilobyte GETVIS for Librarian input override storage; and
- 214-kilobyte GETVIS for ADARAI JCS storage.

When running in z/VSE or z/VM mode on z/VSE hardware, all of the GETVIS and 1 kilobyte of the PSIZE can be run above the 16-megabyte line.

Optional Console or Printer Messages

You have the option of displaying, printing, or preventing these messages by specifying the `JBXEMSG` and `JBXIMSG` parameters in the Adabas options table.

Diagnostic Functions

After the job exit is installed, you can produce dumps of the two tables for diagnostic purposes. Executing SAGINST with the ADASIP UPSI statement:

- UPSI 10000000 produces a dump of the Librarian input override table;
- UPSI 01000000 produces a dump of the ADARAI JCS table.

If the size of these two tables needs to be changed for any reason, SAGIPT may be zapped before being loaded into the SDL:

- The Librarian input override table size may be changed from the default of X‘00014874' (84,084 bytes) to an appropriate value by zapping location X‘18'. When altering the SAGIPT.OBJ module, `ESDID=002` is required on the MSHP AFFECTS statement.
- The ADARAI JCS table size may be changed from the default of X‘000355D6' to an appropriate value by zapping location X‘0C'.

Each element in the Librarian input override table is 42 bytes in length. The default table size assumes 10 SAGUSER statements per file name, 10 file names, and 20 partitions, plus two extra unused entries. This number is an estimate of maximum concurrent residency; each statement is removed from the table after it is used.

Each element in the ADARAI JCS table is 91 bytes in length. The default table size accommodates 2400 entries with each DLBL, TLBL, or EXTENT statement requiring an entry in the table. Whenever a JOB statement is encountered, all entries for that partition (task ID) are cleared from the table.

Acquiring Storage for the ID Table

The SYSTEM GETVIS is used to acquire storage for the ID table (IDT). This storage is acquired using the ADASIP at SVC installation time. The size of storage in the SYSTEM GETVIS depends on the number of IDT entries specified using ADASIP. The default number of IDT entries (IDTEs) is 10. The size can be calculated as follows:

```
SIZE (in bytes) =1024 (IDT prefix) + 96 (IDT header) + (32 x number of IDTEs)
= 1024 + 96 + (32 x 10)
= 1024 + 96 + 320
= 1440 bytes
```

Also, additional SYSTEM GETVIS storage is acquired. This storage permits users to communicate from multiple address spaces when Adabas is not running in a shared partition. In this case, the following formula is used to calculate SYSTEM GETVIS:

```
SIZE (in bytes) = 192 (CQ header) + (192 x NC value) + (4352 x NAB value)
```

It may be necessary to increase the SVA size to meet these requirements. To do so, change the SVA operand in the appropriate $IPL*xxx* procedure, then re-IPL.

Note: By default, the SYSTEM GETVIS is acquired above the 16-megabyte line. To acquire most of this space below the line, linkedit ADARUN AMODE 24.

Acquiring Storage for the IIBS Table

The 31-bit SYSTEM GETVIS is used to acquire storage for the IIBS table (IIBS). This storage is acquired using the ADASIP at SVC installation time. The size of storage in the 31-bit SYSTEM GETVIS is 128K.

SVC Work Areas

For each Adabas SVC installed, a number of 384-byte work areas are reserved. The number of work areas reserved is calculated as four times the number of IDTEs (`4 x IDTE-count`). The maximum number of work areas allocated is 128; the minimum is 24. The SVC work areas therefore occupy between 9K and 48K of storage. The default value of 10 IDTEs results in 15K of SYSTEM GETVIS being allocated.

Displaying Storage Allocation Totals

Specifying `// UPSI xxxxxxxGx` during the ADASIP execution (see UPSI byte description in ***ADASIP Execution Parameters***, earlier in this guide,) will generate allocation messages on the system console, showing the total 24-bit GETVIS and 31-bit GETVIS storage allocated by Adabas:

```
ADASIP85 GETVIS-24 storage allocated: nnnK
ADASIP85 GETVIS-31 storage allocated: nnnK
```

Calls from Other Partitions

In order for an Adabas nucleus to accept calls from other partitions, storage is acquired in the SVA GETVIS area for any required attached buffers. The buffers hold data moved between the nucleus and users in other partitions.

Dummy Sequential Files

If the file is not needed, it can be unassigned or assigned IGN such as the following:

```
// ASSGN SYS014,UA
```

or

```
// ASSGN SYS014,IGN
```

Backward Processing of Tapes and Cartridges

To perform backward processing of tapes or cartridges, file positioning must occur before the file is opened. This can only be done when an assignment is made for the file. When performing the ADARES BACKOUT utility function, the // ASSGN ... for file BACK must be done explicitly.

No tape management system can be used, because such systems perform the assign operation when the file is opened; the LUB and PUB remain unassigned until this occurs.

Applying Zaps (Fixes)

The jobs described in this section can be used to permanently change defaults and apply corrections (zaps) to the libraries in the supported z/VSE systems.

Two methods are used in z/VSE for applying corrective fixes to Adabas:

- the MSHP PATCH facility requires no definition of Adabas as a product/component on the MSHP history file. This method only alters phases. If the phase is relinked, the zap is lost.
- the MSHP CORRECT facility requires the definition of Adabas as a product/component using MSHP ARCHIVE.

Software AG distributes Adabas zaps to z/VSE users in MSHP CORRECT format and therefore recommends that you use MSHP CORRECT.

Applying Fixes Using MSHP PATCH

A sample job for applying a fix to Adabas using MSHP PATCH is as follows:

Note: This sample job is available in member MSHPPAT.X.

```
// JOB PATCH APPLY PATCH TO ADABAS
// OPTION LOG
// EXEC PROC=ADAVvLIB
// EXEC MSHP
PATCH SUBLIB=saglib.ADAvrs
AFFECTS PHASE=phasenam
ALTER offset vvvv : rrrr
/*
/&

where
vrs is the Adabas version
saglib is the Adabas library name in procedure ADAVvFIL
phasenam is the Adabas phase to be zapped
offset is the hexadecimal offset into the phase
vvvv is the verify data for the zap
rrrr is the replace data for the zap
```

Applying Fixes Using MSHP CORRECT

- MSHP ARCHIVE
- MSHP CORRECT

MSHP ARCHIVE

For new users or users with no requirement to maintain multiple versions of Adabas, the following sample job can be used to define Adabas to MSHP.

Note: This job uses the history file identified by the IJSYSHF label in the z/VSE standard label area.

Note: This sample JCL is available in member MSHPARC.X.

```
// JOB ARCHIVE ARCHIVE ADABAS
// OPTION LOG
// EXEC PROC=ADAVvLIB
// EXEC MSHP
ARCHIVE ADAvrs
COMPRISES 9001-ADA-00
RESOLVES 'SOFTWARE AG - ADABAS Vv.r'
ARCHIVE 9001-ADA-00-vrs
RESIDENCE PRODUCT=ADAvrs -
PRODUCTION=saglib.ADAvrs -
GENERATION=saglib.ADAvrs
/*
/&
```

```
—where
vrs is the Adabas version
saglib is the Adabas library name in procedure ADAVvFIL
```

A different MSHP history file must be used for each version and revision level of Adabas to which maintenance is applied.

To preserve the MSHP environment of an older version level of Adabas during an upgrade to a new version, it is necessary to create an additional MSHP history file for use by the new version.

The following sample MSHP job can be used to create an additional history file for a new version of Adabas and define Adabas to it.

Note: This sample JCL is available in member MSHPDEF.X.

```
// JOB ARCHIVE DEFINE HISTORY AND ARCHIVE ADABAS
// OPTION LOG
// EXEC PROC=ADAVvLIB
// ASSGN SYS020,DISK,VOL=volhis,SHR
// EXEC MSHP
CREATE HISTORY SYSTEM
DEFINE HISTORY SYSTEM EXTENT=start:numtrks -
UNIT=SYS020 -
ID='adabas.new.version.history.file'
ARCHIVE ADAvrs
COMPRISES 9001-ADA-00
RESOLVES 'SOFTWARE AG - ADABAS Vv.r'
ARCHIVE 9001-ADA-00-vrs
RESIDENCE PRODUCT=ADAvrs -
PRODUCTION=saglib.ADAvrs -
GENERATION=saglib.ADAvrs
/*
/&

—where
vrs is the Adabas version
volhis is the volume on which the Adabas Vvr history file resides
start is the start of the extent on which the Adabas Vvr history file resides
numtrks is the length of the extent on which the Adabas Vvr history file resides
adabas.new.version.history.file is the physical name of the Adabas Vvr history file
saglib is the Adabas library name in procedure ADAVvFIL
```

Once migration to the new version is complete, you can either

- continue to use the new history file to apply subsequent fixes; or
- delete the old version of Adabas from MSHP and merge the new version into the standard MSHP history file.

Caution: Before running any MSHP REMOVE or MERGE jobs, back up your MSHP environment by running MSHP BACKUP HISTORY jobs against all MSHP history files.

A sample MSHP job to remove an old version of Adabas is provided below.

Note: This sample JCL is available in member MSHPREM.X.

```
// JOB REMOVE REMOVE OLD ADABAS
// OPTION LOG
// PAUSE ENSURE MSHP HISTORY FILE BACKUP HAS BEEN TAKEN
// EXEC MSHP
REMOVE ADAvrs
REMOVE 9001-ADA-00-vrs
/*
/&

—where vrs is the old Adabas version
```

A sample MSHP job to merge an additional history file for Adabas into the standard MSHP history file is provided below.

Note: This sample JCL is available in member MSHPMER.X.

```
// JOB MERGE MERGE SEPARATE ADABAS INTO STANDARD HISTORY
// OPTION LOG
// PAUSE ENSURE MSHP HISTORY FILE BACKUPS HAVE BEEN TAKEN
// ASSGN SYS020,DISK,VOL=volhis,SHR
// EXEC MSHP
MERGE HISTORY AUX SYSTEM
DEFINE HISTORY AUX EXTENT=start:numtrks -
UNIT=SYS020 -
ID='adabas.new.version.history.file'
/*
/&

—where
volhis is the volume on which the Adabas Vvr history file resides
start is the start of the extent on which the Adabas Vvr history file resides
numtrks is the length of the extent on which the Adabas Vvr history file resides
adabas.new.version.history.file is the physical name of the Adabas Vvr history file
```

MSHP CORRECT

The MSHP CORRECT and UNDO jobs use the history file identified by label IJSYSHF in the z/VSE standard label area. If Adabas is maintained from a different MSHP history file, include the following label information in the CORRECT or UNDO job:

```
// DLBL IJSYSHF,'adabas.new.version.history.file'
// EXTENT SYSnnn
// ASSGN SYSnnn,DISK,VOL=volhis,SHR

—where
volhis is the volume on which the Adabas Vvr history file resides
nnn is the user-defined SYS number
adabas.new.version.history.file is the physical name of the Adabas Vvr history file
```

A sample of the use of MSHP CORRECT to install a fix to Adabas is provided below.

Note: This sample JCL is available in member MSHPCOR.X.

```
// JOB CORRECT APPLY ADABAS FIX
// OPTION LOG
// EXEC PROC=ADAVvLIB
// EXEC MSHP
CORRECT 9001-ADA-00-vrs : Axnnnnn
AFFECTS MODE=modname
ALTER offset vvvv : rrrr
INVOLVES LINK=lnkname
/*
/&

—where
vrs is the Adabas version
x is the Adabas component (for example, N for nucleus)
nnnnn is the Adabas fix number
modname is the Adabas object module to be zapped and then relinked
offset is the hexadecimal offset to the beginning of the zap
vvvv is the verify data for the zap
rrrr is the replace data for the zap
lnkname is the link book for the phase affected
```

The CORRECT job updates object and phase in a single job step using the link book feature of MSHP. The INVOLVES LINK= statement automatically invokes the linkage editor after the object module is updated.

For a zap applied with the INVOLVES LINK= statement, the following UNDO can be used to remove the fix from both object module and phase:

Note: This sample JCL is available in member MSHPUND.X.

```
// EXEC MSHP
UNDO 9001-ADA-00-vrs : Axnnnnn
/*
```

where *vrs* is the Adabas *version*, *x* is the Adabas component (for example, N for nucleus), and *nnnnn* is the Adabas fix number.

Adabas provides a link book containing parameters for invoking the linkage editor for each Adabas phase. The name of each link book begins with "LNK" and the member type is "OBJ".

No link book is provided for module ADAOPD or for any other programs distributed in source form. Programs distributed in source form continue to be modified using assembly and link jobs.

If you choose not to take advantage of the link book facility, remove the INVOLVES LINK= statement from any zap before applying it. You can then run the linkage editor step to recreate the phase separately, as before.

This may be done to link a temporary version of a phase into a separate sublibrary for testing purposes. However, it is also possible to maintain a separate test version of Adabas modules by defining an additional z/VSE system history file. See ***Maintaining a Separate Test Environment in z/VSE***.

Link Book Update Requirements for Secondary SVC

If you use the link book facility and require a non-standard SVC suffix (for example, if you relink the Adabas 8.1 SVC to phase ADASVC11), you must remember to update the link book for the SVC (LNKSVC.OBJ) to reflect the new phase name.

The link book provided for ADASVC81 is LNKSVC.OBJ. It contains the following:

```
PHASE ADASVC81,*,NOAUTO,SVA
MODE AMODE(31),RMODE(24)
INCLUDE SVCVSE
INCLUDE SVCCLU
ENTRY ADASVC
```

To set up an SVC with suffix -11, you would need to update the link book as follows:

```
// DLBL SAGLIB,'adabas.Vvrs.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// EXEC LIBR
ACCESS SUBLIB=SAGLIB.ADAvrs
CATALOG LNKSVC.OBJ REPLACE=YES
PHASE ADASVC11,*,NOAUTO,SVA
MODE AMODE(31),RMODE(24)
INCLUDE SVCVSE
INCLUDE SVCCLU
ENTRY ADASVC
/+
/*

—where
vrs is the Adabas version
adabas.Vvrs.library is the physical name of the Adabas vrs library
volser is the volume on which the library resides
```

Link Book Update Requirements for Running AMODE 24

If you use the link book facility and require AMODE 24 versions of any modules linked by default as AMODE 31 (ADARUN, ADASVC74), you must update the corresponding link book (LNK-RUN.OBJ, LNKSVC.OBJ) to remove the MODE statement.

This link book update can be made using a method similar to that described in the previous section for the SVC suffix update.

Adabas 7 Adalink Considerations

User Exit B (Pre-Command) and User Exit A (Post-Command)

One or two user exits may be linked with an Adalink routine:

- UEXITB receives control *before* a command is passed to a target with the router 04 call.

  Note: Special commands emanating from utilities and from Adabas Online System are marked as physical calls. These calls must be bypassed in user exits. These calls have X'04' in the first byte (TYPE field) of the command's Adabas control block (ACB). UEXITB must check this byte and return if it is set to X'04'. Be sure to reset R15 to zero on return.

- UEXITA receives control *after* a command has been completely processed by a target, the router, or by the Adalink itself.

At entry to the exit(s), the registers contain the following:

Register	Contents
1	Address of the UB. If the flag bit UBFINUB is reset, the contents of the halfword at Adabas + X'86' have been moved to UBLUINFO. If those contents are greater than zero, the two bytes starting at UBINFO (UB+X'40') have been set to zero. If UBFINUB is set, no changes can be made to the UB or ACB (except for ACBRSP).
2	Address of a 16-word save area (for ADALNC only)
13	Address of an 18-word save area (for non-CICS Adalink exits)
14	Return address
15	Entry point address: UEXITB or UEXITA

Any registers except register 15 that are modified by the user exits must be saved and restored; the address of a save area for this purpose is in register 13.

If at return from UEXITB register 15 contains a value other than zero (0), the command is not sent to the target but is returned to the caller. The user exit should have set ACBRSP to a non-zero value to indicate to the calling program that it has suppressed the command: response code 216 is reserved for this purpose.

The UEXITB exit may set the UB field UBLUINFO to any lesser value, including zero; an abend occurs if the user exit sets UBLUINFO to a greater value. The UBLUINFO length cannot be changed when any other exit is used.

The user information received by a UEXITA exit may have been modified; this modification may include decreasing its length, possibly to zero, by any of the Adalink user exits.

An Adalink routine can return the following non-zero response codes in ACBRSP:

Response Code	Description
213	No ID table
216	UEXITB suppressed the command
218	No UB available

At least the following three equates, described at the beginning of the source, can be modified before an Adalink routine is assembled. In some Adalink routines, however, the corresponding information can be zapped:

Equate	Description
LOGID	The default logical ID, ranging in value from 1 to 65535. The default is 1.
LNUINFO	The length of the user information to be passed to Adalink user exits, ranging in value from 0 to 32767. The default is 0.
SVCNR	The Adabas SVC number; its range of values and the default depend on the operating system. This value can be provided as SYSPARM value for assembly of the following Adalink routine: `//EXEC PGM=ass,PARM='.....,SYSPARM(svcnr)'`

The first 152 (X'98') bytes of all Adabas Adalinks must maintain the following structure:

Offset	Label	Contents	Meaning
00	ADABAS		Entry code
12		CL6'ADALN*x*'	Program name
18		XL4'*yyyymmdd*'	Assembly date
1C		A(ZAPTAB)	Address of zap table
20	PATCH	XL96'00'	Patch area
80	LNKLOGID	AL2(LOGID)	Default logical ID (default: 1)
82		82 XL2'00'	Reserved
84	LNKSVC	SVC SVCNR	Executable SVC instruction for Adabas SVC (default: operating-system-dependent)
86	LUINFO	Y(LNUINFO)	Length of user information (default: 0)
88	VUEXITA	V(UEXITA)	Address of user exit after call (weak)
8C	VUEXITB	V(UEXITB)	Address of user exit before call (weak)
90	ADABAS51	CL8'ADABAS51'	IDT ID

ADAUSER Considerations

ADAUSER is a program that links the user to Adabas. It is specific to an operating system and is independent of release level and mode. It can be used in batch and in some TP environments.

ADAUSER contains the entry point ADABAS and should be linked with all user programs that call Adabas. No other programs containing the CSECT or entry point name ADABAS can be linked in these load phases.

On the first Adabas call, ADAUSER (CDLOAD) loads the latest version of ADARUN. This makes the calling process release-independent. Subsequent Adabas calls bypass ADARUN.

ADARUN processes its control statements. For the ADARUN setting `PROGRAM=USER` (the default), ADARUN loads the appropriate TP Adalink if the ADARUN parameter setting `MODE=MULTI` is specified, or the Adabas nucleus (ADANUC) if the ADARUN parameter setting `MODE=SINGLE` is specified. This makes the calling process mode independent.

Adabas 8 Adalink Considerations

Link Routine User Exit 1 (Pre-Command) and User Exit 2 (Post-Command)

One or two user exits may be linked with an Adalink routine:

- Link routine user exit 1, LUEXIT1, receives control *before* a command is passed to a target with the router 04 call.

  Note: Special commands emanating from utilities and from Adabas Online System are marked as physical calls. These calls must be bypassed in user exits. These calls have X'04' in the first byte (TYPE field) of the command's Adabas control block (ACBX). LUEXIT1 must check this byte and return if it is set to X'04'. Be sure to reset R15 to zero on return.

- Link routine user exit 2, LUEXIT2, receives control *after* a command has been completely processed by a target, the router, or by the Adalink itself.

At entry to the exit(s), the registers contain the following:

Register	Contents
1	Address of the UB. If the flag bit UBFINUB is reset, the contents of the halfword at Adabas + X'86' have been moved to UBLUINFO. If those contents are greater than zero, the two bytes starting at UBINFO (UB+X'40') have been set to zero. If UBFINUB is set, no changes can be made to the UB or ACB (except for ACBRSP).
2	Address of an 18-word format 1 register save area
13	For CICS, on entry to the link user exit, R13 points to the CICS DFHEISTG work area at *xxxxxxxxx*. For batch/TSO, R13 points to the link routine's work area.
14	Return address
15	Entry point address: LUEXIT1 or LUEXIT2

Any registers except register 15 that are modified by the user exits must be saved and restored; the address of a save area for this purpose is in register 13.

If at return from LUEXIT1, register 15 contains a value other than zero (0), the command is not sent to the target but is returned to the caller. The user exit should have set ACBXRSP to a non-

zero value to indicate to the calling program that it has suppressed the command: response code 216 is reserved for this purpose.

The LUEXIT1 exit may set the UB field UBLUINFO to any lesser value, including zero; an abend occurs if the user exit sets UBLUINFO to a greater value. The UBLUINFO length cannot be changed when any other exit is used.

The user information received by a LUEXIT2 exit may have been modified; this modification may include decreasing its length, possibly to zero, by any of the Adalink user exits.

An Adalink routine can return the following non-zero response codes in ACBXRSP:

Response Code	Description
213	No ID table
216	LUEXIT1 suppressed the command
218	No UB available

LNKUES for Data Conversion

The Adabas 8 standard batch ADALNK is delivered with UES (Universal Encoding Support). The LNKUES module, as well as the modules ASC2EBC and EBC2ASC, are linked into the standard batch ADALNK. LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller.

LNKUES is called only on ADALNK request (X'1C') and reply (X'20') calls if the first byte of the communication ID contains X'01' and the second byte does not have the EBCDIC (X'04') bit set.

- For requests, LNKUES receives control before LUEXIT1.
- For replies, LNKUES receives control after LUEXIT2.

By default, two translation tables are linked into LNKUES/ADALNK:

- ASC2EBC: ASCII to EBCDIC translation; and
- EBC2ASC: EBCDIC to ASCII translation.

Note: It should only be necessary to modify these translation tables in the rare case that some country-specific character other than "A-Z a-z 0-9" must be used in the Additions 1 (user ID) or Additions 3 field of the control block.

If you prefer to use the same translation tables that are used in Entire Net-Work:

- In ASC2EBC and EBC2ASC, change the COPY statements from UES2ASC and UES2EBC to NW2ASC and NW2EBC, respectively.
- Re-assemble the translation tables and re-link LNKUES/ADALNK.

Both the Adabas and Entire Net-Work translation table pairs are provided in the section ***Translation Tables***. You may want to modify the translation tables or create your own translation table pair. Be sure to (re)assemble the translation tables and (re)link LNKUES/ADALNK.

The following is a sample job for (re)linking ADALNK with LNKUES and the translation tables:

```
*
// JOB ...
// EXEC PROC=
// LIBDEF *,SEARCH=(search-chain-library.sublib ...)
// LIBDEF PHASE,CATALOG=(lib.sublib)
PHASE ADALNK,*
MODE AMODE(31),RMODE(24)
INCLUDE LNKVSE8
INCLUDE LINKIND
INCLUDE LINKGBLS
INCLUDE LNKUES
INCLUDE ASC2EBC
INCLUDE EBC2ASC
ENTRY ADABAS
// EXEC LNKEDT
```

The (re)linked ADALNK must be made available to Entire Net-Work. If you are calling Adabas 8 and you do not have the correct LNKUES/ADALNK module, Adabas produces unexpected results: response code 022, 253, etc.

ADAUSER Considerations

ADAUSER is a program that links the user to Adabas. It is specific to an operating system and is independent of release level and mode. It can be used in batch and in some TP environments.

ADAUSER contains the entry point ADABAS and should be linked with all user programs that call Adabas. No other programs containing the CSECT or entry point name ADABAS can be linked in these load phases.

On the first Adabas call, ADAUSER (CDLOAD) loads the latest version of ADARUN. This makes the calling process release-independent. Subsequent Adabas calls bypass ADARUN.

ADARUN processes its control statements. For the ADARUN setting `PROGRAM=USER` (the default), ADARUN loads the non-reentrant Adalink modules. To load a reentrant batch link routine, use the ADARUN parameter `PROGRAM=RENTUSER`. This makes the calling process mode-independent.

Setting Defaults in ADARUN

The member DEFAULTS.X is available for setting the ADARUN defaults.

DEFAULTS.X uses MSHP CORRECT to install the fix.

Default Name	Current Value
Device type	3390
SVC number	45
Database ID	1

4 Installing Adabas With TP Monitors

This chapter provides information needed to install Adabas in batch mode and with its teleprocessing (TP) monitors.

Preparing Adabas Link Routines for z/VSE

This section covers the following topics:

High-Level Assembler

IBM has dropped support for the old z/VSE assembler and Software AG supports assembling the Adabas 7 link components with the high-level assembler only.

Addressing Mode Assembly Directives

The Adabas link routines now have `AMODE` and `RMODE` assembly directives in the source. These allow the linkage editor to produce warning messages when conflicting `AMODE` or `RMODE` linkage editor control statements are encountered in the link JCS or EXECs.

These assembly directives also serve to document the preferred `AMODE` and `RMODE` for each link routine. It is important to note that in and of themselves, these directives do not alter the actual addressing mode of the link routine during execution.

The batch link routine ADALNK has the following `AMODE` and `RMODE` assembly directives:

```
ADABAS AMODE 31
ADABAS RMODE 24
```

Software AG recommends `RMODE 24` for the z/VSE non-reentrant batch link routine (ADALNK.PHASE).

Modifying the Assembly Directives

These directives may be changed by modifying the source members before assembling them, or they may be overridden by linkage editor control statements. For example, to link the batch ADALNK module with `AMODE31` and an `RMODE ANY`, the following control statements may be provided as input to the linkage editor:

```
PHASE ADALNK,*
MODE AMODE(31),RMODE(ANY)
```

The linkage editor control statements override the Assembler directives in the source module.

For more information about the AMODE and RMODE directives and their effects on the assembler, linkage editor, and execution, consult IBM's *VSE Extended Addressability Guide*.

Re-linking Adabas 8 Link Routines

When re-linking the Adabas 8 link routines with certain AMODE and RMODE combinations, a warning message may be generated by the linkage editor. This may be safely ignored as long as it pertains to a conflict of AMODE or RMODE in the ESD record of one or more of the load modules that comprise the link routine, and as long as the resulting module has the proper AMODE and RMODE attributes for execution with the intended calling application programs.

Care must be taken to ensure that AMODE(24) applications will operate properly when invoking the link routine with the attributes chosen when it is re-linked. This is particularly important if the RMODE(ANY) attribute is associated with a link routine that will be loaded dynamically but invoked by a program that is AMODE(24). In this case, the link routine should be re-linked AMODE(31),RMODE(24) to avoid addressing exception ABENDs because the AMODE(24) application cannot correctly invoke the link routine if it resides above the 16-megabyte line.

The Adabas 8 link routines all run AMODE(31) after initialization, but they will return to the caller in the caller's AMODE.

Note: Under CICS, the V8 links run AMODE(31), but the Dataloc RDO parameter governs the AMODE and RMODE of the running CICS transaction.

The batch z/VSE link routine, ADALNK, has been assembled and link-edited AMODE(31,RMODE(24). This provides the most flexible configuration for most z/VSE applications that will invoke it. It may be re-linked AMODE(31),RMODE(ANY), but you must be certain that no AMODE(24) applications will invoke it.

The reentrant batch link routine, ADALNKR, has been assembled AMODE(31),RMODE(24). It may be re-linked AMODE(31),RMODE(ANY) if no AMODE(24) applications will invoke it.

The z/VSE Com-plete link routine, ADALCO, has been assembled and link-edited AMODE(31),RMODE(24), and this is the required configuration for ADALCO under z/VSE Com-plete because ADALCO still uses z/VSE macros and services which require it to reside below the 16-megabyte line.

All of the Adabas 8 CICS link routine modules - ADACICS, ADACICT, ADACICO, and ADACIRQ - have been assembled and link-edited AMODE(31),RMODE(ANY). CICS manages the loading of programs and their invocation depending on the DATALOC values associated with their program and transaction definitions.

ADAUSER AMODE/RMODE Considerations

Software AG recommends that all batch applications invoke Adabas calls through the ADAUSER module. This module is normally link-edited with the application program and it then loads the appropriate link routine as well as ADARUN and ADAIOR/ADAIOS. The source member has the `AMODE` and `RMODE` directives coded as `AMODE 31`, `RMODE ANY`. This is the most flexible configuration for assembling and linking ADAUSER with the widest variety of application programs. However, if ADAUSER is dynamically loaded, either the `RMODE` assembler directive should be changed to `RMODE 24` before re-assembling it or the ADAUSER module should be re-linked `AMODE(31),RMODE(24)` to ensure that `AMODE 24` application programs may invoke it properly below the 16-megabyte line.

UES-Enabled Link Routines

For prior versions of Adabas, UES is enabled by default for only the batch and Com-plete link routines. As of Adabas version 8, UES is enabled by default for *all* link routines, including the CICS link routines. It is not necessary to disable UES support. Applications that do not require UES translation continue to work properly even when the UES components are linked with the Adabas link routines. See the section ***Enabling Universal Encoding Support (UES) for Your Adabas Nucleus*** for more information.

This section covers the following topics:

Default or Customized Translation Tables

By default, the load modules for all Adabas 8 link routines have been linked with LNKUES and the default translation tables.

LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller.

The two standard translation tables are:

- ASC2EBC: ASCII to EBCDIC translation; and
- EBC2ASC: EBCDIC to ASCII translation.

The Adabas translation table pair is provided in the section ***Translation Tables***.

You may use the load modules with the default translation tables linked in, or you may prepare your own customized translation tables, re-assemble the tables, and link them with the LNKUES module that is delivered.

Notes:

1. It should only be necessary to modify these translation tables in the rare case that some country-specific character other than "A-Z a-z 0-9" must be used in the Additions 1 (user ID) or Additions 3 field of the control block.
2. The load module LNKUESL delivered with earlier levels of Adabas Version 7 is no longer supplied since the link jobs now specify the LNKUES or LNKUES7 module and the translation tables separately.
3. The LNKUES module is functionally reentrant; however, they is not linked that way in the Adabas load library.
4. When linking the LNKUES load module and the translation tables, the linkage editor may produce warning messages concerning the reentrant or reusability status of the linked module. These warning messages can be ignored.
5. If relinking an Adabas 8 link routine for UES support, the LNKUES module must be included. This will ensure that your new Adabas 8 applications have support for Adabas 8 direct calls and control blocks.

Calling LNKUES

LNKUES is called only on Adabas link routine request (X'1C') and reply (X'20') calls if the first byte of the communication ID contains X'01' and the second byte does not have the EBCDIC (X'04') bit set. In Adabas 8 requests, LNKUES receives control before UEXIT1. In Adabas 8 replies, LNKUES receives control after UEXIT2.

Adabas 8 Jobs for z/VSE Universal Encoding Support

The following lists the sample jobs provided to manage universal encoding support in Adabas link routines in z/VSE environments:

Sample Job	Description
ALNKCIC8.X	Assembles and links the CICS globals table with LNKUES and the default translation tables ASC2EBC and EBC2ASC.
ALNKLCO8.X	Relinks the Com-plete link routine with the LCOGBL link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.
ALNKLNK8.X	Relinks the batch link routine with the LNKGBLS link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.
ALNKLNR8.X	Relinks the reentrant batch link routine with the LNKRGBL link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.

Before you can use any of these jobs, they should be edited to prepare the job power statements, provide the proper names for the procedures and libraries referenced, and all necessary extent and volume information. Refer to the comments in the jobs themselves for more information.

Disabling UES Support for Adabas 8 Link Routines

This section describes how to disable UES support in the Adabas 8 Com-plete and batch link routines, if for some reason you feel it is necessary.

To disable UES support in link routines:

1. Edit the link globals table for the associated link routine. Set the UES parameter setting to NO.
2. Assemble the link globals table after making any other necessary modifications to any other keyword directives in the source module as required by your installation.
3. Link the Adabas link routine with the newly assembled link globals table and do not include any of the UES components (that is, LNKUES, ASC2EBC, or EBC2ASC).

For more information about the specific link routines, read *Installing Adabas with Com-plete under Adabas 8*, and *Installing Adabas with Batch under Adabas 8*, elsewhere in this guide.

General Considerations for Installing Adabas with CICS

The Adabas command-level link routine supports the CICS transaction server (TS) 1.1 running under z/VSE 2.4 and above.

How Adabas is installed on CICS-based systems depends on the level of CICS being run:

- The command-level link from Adabas 8.1 cannot be used with CICS/VSE 2.3. Instead, you must use the command-level link routine for Adabas Version 7.4.4 or the macro-level link routine provided in source in the 7.4.4 VSE sublibrary with CICS/VSE 2.3 environments.
- CICS TS 1.1 running under z/VSE 2.4 and above must run a current version of Adabas and use the command-level link component.

Note: The OPID option for the USERID field is not supported under CICS/VS 2.3 and above; therefore, it is not provided with the command-level link routine.

This section covers the following topics:

CICS Release Support

IBM has not announced a date for end of service for CICS/VSE 2.3. Consequently, Software AG will continue to support the Adabas CICS 7.4.4 link routines, particularly the macro-level ADALNC routine on systems running CICS/VSE 2.3 until IBM drops support for that level of CICS.

For CICS/TS 1.1 and above for VSE, the Adabas 8.1.4 CICS link components are supported and the Adabas CICS 7.4 link routines will not be supported when general support for Adabas 7.4.4 is terminated by Software AG.

CICS MRO Environment Requirements

If you run the Adabas CICS command-level link routine with the CICS multiple region option (MRO), you must set the ADAGSET option `MRO=YES` and use the default value for the ADAGSET `NETOPT` option.

You can use the ADAGSET `NTGPID` option to provide a 4-byte literal for the Adabas communication ID to be used by the Adabas SVC when applications that call Adabas span multiple application regions.

Alternatively, you can create a user exit B (UEXITB) for the link routine that

- sets UBFLAG1 (byte X'29' in the UB DSECT) to a value of X'08' (UBF1IMSR); and
- places a 4-byte alphanumeric value in the UB field UBIMSID.

The exit then allows the Adabas SVC to provide a proper Adabas communication ID in the Adabas command queue element (CQE) even when transactions originate in multiple regions.

Sample Resource Definitions

Under CICS/TS 1.1 and above for z/OS and z/VSE, the preferred method for defining and installing CICS programs and transactions is RDO (resource definition online). The CICS documentation no longer recommends the assembly of PPT and PCT entries to define resources.

The following table provides sample RDO definitions for the Adabas CICS command-level link components. The data has been extracted directly from the CICS CSD file and should be used as a guide for providing comparable information on the CEDA panels.

```
*********************************************************************
* Sample DEFINE control statements for the DFHCSDUP utility.
* For Adabas CICS command-level link routine components.
*
* These control statements can be used as input to the DFHCSDUP
* CICS CSD update utility to define the Adabas CICS command-level
* link routine components on a CICS/TS system.
*********************************************************************
DEFINE PROGRAM(ADABAS) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s COMMAND LEVEL LINK ROUTINE)
LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE PROGRAM(ADACICO) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s PLTPI ENABLE ADATRUE PROGRAM)
LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE PROGRAM(ADATEST) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s DISPLAY GWA PROGRAM - DISPGWA)
STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE PROGRAM(ADATRUE) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s TASK RELATED USER EXIT)
LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(CICS) EXECUTIONSET(FULLAPI)

DEFINE TRANSACTION(DGWA) GROUP(ADABAS)
DESCRIPTION(TRANSACTION TO DISPLAY ADABAS GWA)
PROGRAM(ADATEST) TWASIZE(128) PROFILE(DFHCICST) STATUS(ENABLED)
TASKDATALOC(ANY) TASKDATAKEY(CICS) STORAGECLEAR(NO)
RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) DYNAMIC(NO)
PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO) INDOUBT(BACKOUT)
RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
RESSEC(NO) CMDSEC(NO)
```

—where *s* is the system maintenance level of Adabas.

These sample DEFINE statements are located in member DEFADA7 in the Adabas sublibrary. They can be modified and used as input to the IBM DFHCSDUP utility to define the Adabas CICS command-level components. Consult the appropriate IBM CICS documentation for information on the DFHCSDUP utility.

Installing Adabas with CICS under Adabas 8

Supplied Modules

The following table lists the modules supplied in your Adabas installation to support the installation of Adabas 8 with CICS TP monitors.

Note: The Adabas 8 installation supports Adabas 7 direct calls in addition to Adabas 8 calls; however, an Adabas 7 installation does not support Adabas 8 direct calls.

Module	Description
ADACIC0.OBJ	CICS initialization program code object module.
ADACIC0.PHASE	CICS initialization executable module.
ADACICS.OBJ	CICS TP monitor program code object module. This module is linked with ADADCI.OBJ to produce ADACICS.PHASE.
ADACICS.PHASE	CICS TP monitor executable module.
ADACICT.OBJ	CICS task-related user exit (TRUE) program code object module, dependent part. This module is linked with LNKCIM.OBJ to produce ADACICT.PHASE.
ADACICT.PHASE	CICS TRUE executable module.
ADACIRQ.OBJ	Extra-partition transient data queue handler program code object module.
ADACIRQ.PHASE	Extra-partition transient data queue handler object module executable module.
ADADCI.OBJ	Direct call interface program code object module. This module is linked with ADACICS.OBJ to produce ADACICS.PHASE.
CICSGBL.A	Sample link globals table. This module is modifiable. Once it is modified, you can use the ALNKCIC8.X sample JCS to assemble the CICSGBL.A module, producing the CICSGBL.OBJ object module and then link-editing all relevant CICS program code object modules to create the relevant CICS phases required for Adabas 8 support.
CICSGBL.OBJ	Link globals table object module.
CICSGBL.PHASE	Link globals table executable module. This module is identified via DDLINK.
LNKCIC0.OBJ	CICS link book used when applying maintenance with MSHP to link-edit ADACIC0.OBJ to produce ADACIC0.PHASE.
LNKCICG.OBJ	CICS link book used when applying maintenance with MSHP to link-edit the CICSGBL.OBJ globals table and produce CICSGBL.PHASE.
LNKCICS.OBJ	CICS link book used when applying maintenance with MSHP to link-edit ADADCI.OBJ and ADACICS.OBJ to produce ADACICS.PHASE.

Module	Description
LNKCICT.OBJ	CICS link book used when applying maintenance with MSHP to link-edit ADACICT.OBJ and LNKCIM.OBJ to produce ADACICT.PHASE.
LNKCIM.OBJ	CICS task-related user exit (TRUE) product code object module, independent part. This module is linked with ADACICT.OBJ to produce ADACICT.PHASE.
LNKCIRQ.OBJ	CICS link book used when applying maintenance with MSHP to link-edit ADACIRQ.OBJ to produce ADACIRQ.PHASE.

Installation Procedure Under Adabas 8

To install the Adabas 8 CICS link routine components, complete the following steps:

1 Modify the CICS startup JCS to include the Adabas 8 sublibrary in the LIBDEF chain.

2 Modify the sample CICSGBL.A member found in the Adabas 8 sublibrary. This member contains sample default installation (LGBLSET) parameter settings. For more information about what to modify in this member, read *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*, elsewhere in this section.

Note: The setting for the `OPSYS` parameter must be z/VSE.

3 Save the modified CICSGBL.A member with a unique name in an appropriate user sublibrary.

4 Using sample job ALNKCIC8.X, assemble and link-edit the member you saved in the previous step into a sublibrary that will be made available to CICS in the LIBDEF concatenation. Note that any user or Software AG link routine exits should be link-edited with this load module. (For information about specific Software AG product exits, read the installation documentation for the product.)

5 Modify the DEFADA8.A member to provide the correct name of the link routine globals default table created in the previous step (Step 4). The default module name is CICSGBL. Tailor this member for any other CICS installation values as required.

Note: The Adabas 8 CICS program names, other than the name of the link globals table, are predefined and cannot be changed (for example, ADACICS, ADACICT, ADACIRQ, and ADACIC0).

6 Run the IBM DFHCSDUP utility to update the CICS CSD file for the desired CICS using the modified DEFADA8 member as input.

7 Modify the CICS PLTPI table to add the entries that will enable and start the Adabas CICS task-related user exits (TRUE). Use member ADAPLTXX.A from the Adabas 8 sublibrary as a sample for enabling and starting a legacy Adabas TRUE and the new Version 8 TRUE in the second phase of the PLT.

8 Assemble and link the modified PLTPI table into a library that will be available to the desired CICS region.

9 Update a Destination Control Table (DCT) to include the entries found in member DCTAV8.A in the Adabas 8 sublibrary. Assemble and link-edit this table with a unique suffix into a sub-library that will be made available to CICS. Modify the CICS SIT parameters to reference the updated DCT.

10 Modify the CICS startup JCL to include the DDLINK DLBL and EXTENT statements that provide the name of the link globals table prepared in Step 4. For more information on doing this, read ***Preparing DDLINK Input for CICS***, elsewhere in this section.

11 Start the CICS and note any messages relating to the installation of the Adabas TRUE modules that appear on the console.

Preparing DDLINK Input for CICS

Operation of the Adabas 8 CICS link routines may be tailored for each CICS partition by assembling and linking a link globals defaults table and making that table available to CICS at execution. The table may have any legal phase name that is acceptable to CICS and that does not conflict with existing program names used in the CICS region.

The globals table must be defined to CICS as a program. Review the provided sample DEFADA8.A member found in the Adabas 8 sublibrary to see a definition of the sample called CICSGBL. The DEFADA8.A member should be modified as necessary and provided as input to the DFHCSDUP utility to define the Adabas 8 components in the CICS CSD. This process is also described as part of the installation procedure under Adabas 8. Consult the IBM CICS documentation for information on the DFHCSDUP utility.

Each link routine globals table for a CICS region may have a unique name. The Adabas 8 CICS link routines are provided this name through an extra-partition transient data queue. The queue-name is ADAI and its definition is provided in the sample DCT macro definitions found in member DCTAV8.A. Copy this member into a complete DCT member intended for the CICS. The DCTAV8.A member contains the following DFHDCT definitions:

```
DFHDCT TYPE=SDSCI, +
DSCNAME=DDLINK, +
DEVICE=DISK, +
RECFORM=FIXUNB, +
RECSIZE=80, +
BLKSIZE=80, +
TYPEFLE=INPUT
DFHDCT TYPE=EXTRA, +
DSCNAME=DDLINK, +
DESTID=ADAI
```

Assemble and link-edit the modified DCT member with a unique suffix (for example, DFHDCT*xx* where *xx* is the unique suffix). Modify the CICS SIT DCT parameter to provide the suffixe, DCT=*xx*.

When the Adabas 8 task-related user exit (TRUE) is enabled via the ADACIC0 program, either at CICS startup or with the ADA0 transaction, the ADACIRQ module is invoked and reads the ADAI transient data queue. The data read is provided in a file. The CICS JCS must be modified to provide this file's DLBL/EXTENT information or the default link globals table name "CICSGBL" will be used. If no link globals table is located, the Adabas 8 TRUE will not be enabled and started.

The format of the input data to be read is:

Content	Description
ADALNK	This keyword must appear in columns 1 through 6.
space	A blank space must appear in column 7.
LGTNAME= or LGT=	The keyword LGTNAME or LGT, followed by an equals sign (=) must appear after the space in column 7, starting in column 8.
module-name	The module name of the prepared link globals table must appear after the equal sign that follows the LGTNAME or LGT keyword.

For example, the following might be placed in a file:

```
ADALNK LGTNAME=LNKCI02
```

In this example, the link default globals table named "LNKCI02" must be prepared, assembled, link-edited, and defined to this CICS.

Preparing the DDLINK Data for Adabas CICS

The Adabas 8 z/VSE CICS link routine installation program, ADACIC0 invokes the ADACIRQ program to read the DDLINK extra-partition file. This file contains the card that provides the name of the link globals table to use for the CICS where the link routines are being installed. The Adabas installation sublibrary contains member DDLINK.X as a sample card for this purpose. This member may be modified to provide the name of the link globals table of your choice.

This data needs to be placed in a SAM file so it may be read by the ADACIRQ program. This can be done in at least three ways:

1. Use the OBJMAINT program to copy a card from SYSIPT to a disk file. Sample job CPYDDL.X may be modified for this purpose.
2. Use LIBR to punch a modified version of the DDLINK.X member to a disk file. Tailor a JCS member to assign SYSPCH to the disk file and punch the member from a sublibrary using the FORMAT=NOHEADER keyword on the PUNCH statement. The control statements should be:

```
// EXEC LIBR,PARM='MSHP'
  ACCESS SUBLIB=SAGLIB.ADAvrs
  PUNCH DDLINK.X   FORMAT=NOHEADER
/*
```

3. Use DITTO to copy the card image from SYSIPT to a disk file.

   Once the disk file has been created with the card image, the CICS execution JCS should be modified to add the following DLBL and extent cards:

```
// DLBL DDLINK,'ADABAS.ADA8.DDLINK',0,SD
// EXTENT SYSnnn,vvvvvv
ASSN SYSnnn,DISK,VOL=vvvvvv,SHR
 ↩
```

Installing Adabas with Com-plete under Adabas 8

The following table lists the modules supplied in your Adabas installation to support the installation of Adabas with Com-plete TP monitors.

Note: The Adabas 8 installation supports Adabas 7 direct calls in addition to Adabas 8 calls; however, an Adabas 7 installation does not support Adabas 8 direct calls.

Supplied Module	Description
ADALCO.PHASE	Com-plete TP monitor executable module.
LCOGBL.A	Link globals table source module. This module is modifiable. Once it is modified, you can use sample job ALNKLCO8.X to assemble an LCOGBL.OBJ module . This sample job also link-edits the LCOGBL.OBJ module with LCOVSE8.OBJ to produce ADALCO.PHASE.
LCOGBL.OBJ	Link globals table object module assembled from LCOGBL.A.
LCOVSE8.OBJ	Com-plete TP monitor program code object module.
LNKLCO.OBJ	Com-plete link book containing link-edit control cards used when applying maintenance with MSHP to link-edit LCOGBL.OBJ and LCOVSE8.OBJ to produce ADALCO.PHASE.

Certain Adabas parameters are required by Com-plete, Software AG's TP monitor, when installing Adabas. For more information, see the *Com-plete System Programmer's* manual.

Software AG's TP monitor, Com-plete, requires an Adabas link routine if it is to communicate with Adabas databases, use Software AG's Entire Net-Work product, or use products like Entire System Server running under Com-plete. At this time, Com-plete does not support a mixed Adabas 7 and Adabas 8 link routine environment; thus Com-plete must be run with either an Adabas 7 link routine or an Adabas 8 link routine.

The Adabas Version 8 link routine is delivered in member ADALCO of the Adabas 8 sublibrary. This member must be linked with a link globals module you prepare and with any link routine exits you require to create the final ADALCO load module that is loaded by Com-plete when Com-plete is initialized. The final ADALCO load module and any exits linked with it must be reentrant.

To prepare the Adabas 8 link routine:

1 Edit the LCOGBL.A member in the Adabas 8 distribution sublibrary. LCOGBL.A is a module containing LGBLSET parameters that are used to create default settings for Com-plete link components. A complete description of LGBLSET parameters can be found in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*, elsewhere in this guide.

Note: The OPSYS parameter must be set to z/VSE.

2 Modify and run the ALNKLCO8.X member to assemble and link-edit the link globals table you updated in the previous step.

The ALNKLCO8.X member will assemble and catalog the link globals table for Com-plete and link it with the Com-plete link routine, LCOVSE8.OBJ and any required exits. The ALNKLCO8.X member provides link-edit control cards for the inclusion of the Adabas 8 LNKUES module with the ASC2EBC and EBC2ASC translation tables.

3 Place the final phase, ADALCO, in a library that will be part of the Com-plete LIBDEF search chain.

Note: The defaults set in the link globals table for Com-plete are primarily for documentation purposes. The Adabas/Com-plete interface module, TLOPADAB, sets values for Adabas target ID and SVC number on each Adabas call. However, it is necessary to include the link globals table object module and any necessary exits, including user exits when linking the Adabas 8 ADALCO.PHASE. If user exits are to be linked with ADALCO, be certain to code the LGBLSET keywords accordingly.

The Adabas 8 link routine is prepared.

Installing Adabas with Batch under Adabas 8

ADALNK is the standard Adalink for running Adabas in batch. ADALNKR (LNKVSER) is supplied as a reentrant batch link routine.

Batch applications should be linked with the ADAUSER module to provide the greatest degree of application calling isolation when invoking the Adabas batch link routines. The ADAUSER module will provide code to load the appropriate link routine and the supporting ADARUN and ADAIOR modules. ADARUN, in turn, loads other modules. To start a user program linked with

ADAUSER, the following modules must be available in the LIBDEF search chain: ADAIOR, ADAIOS, ADALNK, ADAMLF, ADAOPD, ADAPRF, and ADARUN. In addition, ADAUSER reads DDCARD input from SYSIPT or DISK to allow jobstep setting of the database ID, Adabas SVC number, and other parameters.

For non-reentrant operation, the DDCARD input should provide the keyword PROG=USER. This causes ADARUN to load ADALNK for non-reentrant batch operations.

If you want to use reentrant batch operations, the ADAUSER module can still be linked with the application program, but the PROG=RENTUSER keyword must be coded on the DDCARD input. ADAUSER is, however, non-reentrant. For full reentrant batch applications, it will either need to be loaded (CDLOAD) separately, or the ADALNKR.PHASE must be loaded without using the ADAUSER module. In this case, the default values for DBID, SVC number, length of user information, and which exits are to be used is provided by the linked link globals table, as modified (read ***Installing the Reentrant Batch z/VSE Adabas 8 Link Routine***, elsewhere in this section. It is also possible to zap the ADALNKR.PHASE or LNKVSER8.OBJ module with these defaults, but Software AG recommends coding and linking the link globals table instead. Additional information on using a reentrant batch link routine is also provided in *Required Application Reentrancy Propertie* in *&adamf_op;*.

This section covers the following topics:

- Supplied Modules
- Installing the Batch z/VSE Adabas 8 Link Routine
- Installing the Reentrant Batch z/VSE Adabas 8 Link Routine

Supplied Modules

The following table lists the modules supplied in your Adabas installation to support the installation of Adabas 8 with batch.

Note: The Adabas 8 installation supports Adabas 7 direct calls in addition to Adabas 8 calls; however, an Adabas 7 installation does not support Adabas 8 direct calls.

Module	Description
ADALNK.PHASE	Batch executable module.
ADALNKR.PHASE	Batch reentrant executable module.
LNKGBLS.A	Batch link globals table. This module is modifiable. Once it is modified, you can use the ALNKLCO8.X sample JCS to assemble the LNKGBLS.A module, producing the LNKGBLS.OBJ module and then link-editing the LNKGBLS.OBJ module with the LNKVSE8.OBJ module to create the ADALNK.PHASE.
LNKGBLS.OBJ	Batch link globals table object module assembled from LNKGBLS.A.
LNKRGBL.A	Batch reentrant link globals table. This module is modifiable. Once it is modified, you can use the ALNKLNR8.X sample JCS to assemble the LNKRGBL.A module, producing

Module	Description
	the LNKRGBL.OBJ module and then link-editing the LNKRGBL.OBJ module with the LNKVSER8.OBJ module to create the ADALNKR.PHASE.
LNKRGBL.OBJ	Batch reentrant link globals table object module assembled from LNKRGBL.A.
LNKLNK.OBJ	Batch link book containing link-edit control cards used when applying maintenance with MSHP to link-edit LNKGBLS.OBJ and LNKVSE8.OBJ modules to produce ADALNK.PHASE.
LNKLNKR.OBJ	Batch link book containing link-edit control cards used when applying maintenance with MSHP to link-edit reentrant LNKRGBL.OBJ and LNKVSER8.OBJ modules to produce ADALNKR.PHASE.
LNKVSE8.OBJ	Batch program code object module.
LNKVSER8.OBJ	Batch reentrant program code object module.

Installing the Batch z/VSE Adabas 8 Link Routine

To install the Adabas 8 non-reentrant link routine for z/VSE batch, complete the following steps:

1 Edit member LNKGBLS.A in the Adabas distribution sublibrary. Provide values for the LOGID, SVC, GBLNAME, and other keywords to suit your installation requirements. This module contains LGBLSET parameters used to create default settings for link components. A complete description of LGBLSET parameters can be found in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*, elsewhere in this guide.

Note: The parameter OPSYS should be set to z/VSE.

2 Edit the ALNKLNK8.X member found in the Adabas 8 sublibrary. This member will assemble and catalog the LNKGBLS.A module and link it and any desired exits with the LNKVSE8.OBJ module to create the ADALNK.PHASE member for Adabas 8. The ALNKLNK8.X member includes sample link-edit control cards to support UES by including the LNKUES.OBJ. module with the ASC2EBC and EBC2ASC translation tables. Modify the link-edit control cards to include any additional Software AG exit or user exit, as specified in the udpated LNKGBLS.A member.

3 Provide the ADALNK.PHASE member in the LIBDEF search chain for the jobstep that will require Adabas database access or Software AG services.

Installing the Reentrant Batch z/VSE Adabas 8 Link Routine

To install the Adabas 8 reentrant link routine for z/VSE batch, complete the following steps:

1 Edit member LNKRGBL.A in the Adabas distribution sublibrary. Provide values for the LOGID, SVC, GBLNAME, and other keywords to suit your installation requirements. This module contains LGBLSET parameters used to create default settings for link components. A complete description of LGBLSET parameters can be found in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*, elsewhere in this guide.

Note: The parameter OPSYS should be set to z/VSE.

2 Edit the ALNKLNR8.X member found in the Adabas 8 sublibrary. This member will assemble and catalog the LNKRGBL.A module and link it and any desired exits with the LNKVSER8.OBJ module to create the ADALNKR.PHASE member for Adabas 8. The ALNKLNR8.X member includes sample link-edit control cards to support UES by including the LNKUES.OBJ. module with the ASC2EBC and EBC2ASC translation tables. Modify the link-edit control cards to include any additional Software AG exit or user exit, as specified in the udpated LNKRGBL.A member.

3 Provide the ADALNKR.PHASE member in the LIBDEF search chain for the jobstep that will require Adabas database access or Software AG services.

Establishing Adabas SVC Routing by Adabas Database ID

Your application programs that use Adabas link routines in z/OS and VSE environments can route database calls through specific Adabas SVCs, based on the database ID used in the call. SVC routing is managed through the use of a DBID/SVC routing table you supply. Up to 1000 database IDs may be specified in the table and associated with any number of valid SVC numbers installed in the z/OS or VSE system. The DBID/SVC routing table is created using the MDBSVC macro.

Duplicate database IDs are not allowed in the DBID/SVC routing table as there is no reliable way for the link routine to determine which SVC should be used for a database ID if it is listed more than once. If duplicate database IDs are found while the table is being assembled, they are flagged with an assembler MNOTE and a return code of 16 is returned for the assembly attempt.

Notes:

1. Adabas client-based add-ons, such as Adabas Transaction Manager, are not compatible with this feature since for client-based functionality to work, it must be channeled through only a single router for any given session, not across routers. To avoid problems if the dynamic SVC by DBID routing feature in enabled for these products, error messages are issued, the assembly

step of the globals table will receive return code 16, and the globals table load module will not be generated.

2. ADALNK linked with the ADASVCTB should only be used by application programs and should not be made available to the Adabas nucleus or to Entire Net-Work.

Caution: This feature should be used with caution. Transactional integrity is not guaranteed. If an application makes calls to multiple databases that are routed to more than one Adabas SVC, it becomes possible to issue ET, BT, OP, CL, RC, or other Adabas commands that may affect the transaction on one database, but not on the other databases running on different Adabas SVCs that were accessed previously. It therefore is the responsibility of the application program to ensure that all necessary logic is included to ensure transactional integrity across multiple databases where multiple Adabas SVCs are employed.

This section covers the following topics:

Installing the Adabas DBID/SVC Routing Feature

The general steps for installing the Adabas DBID/SVC routing feature are:

1. Define the DBID/SVC routing table in a library member using MDBSVC macro statements. For more information about the DBID/SVC routing table and the MDBSVC macro, read ***Using the MDBSVC Macro***, elsewhere in this section.
2. Assemble and link-edit the DBID/SVC routing table member to create a load module or PHASE that will be made available to the operating environment where the SVC routing feature will be used.
3. Modify a link globals table for the operating environment, specifying the LGBLSET keywords DYNDBSVC=YES and DBSVCTN=*name*, where *name* is the name of the DBID/SVC routing table load module that should be used by the link routine. Assemble and link-edit the updated link globals table as required for the operating environment. For more information about the link globals table and the LGBLSET macro, read ***Modifying Source Member Defaults (LGBLSET Macro) in Version 8*** , elsewhere in this guide. For information on assembling and link-editing the link globals table once the table is updated, refer to the instructions for each z/OS or VSE TP monitoring environment, provided elsewhere in this section.
4. Make the prepared DBID/SVC routing table available in a load library that is accessible by the application program's job step, so it can be loaded by the link routine when it runs.
5. Except for CICS systems, you will need to relink ADALNK or ADALNKR making sure that the INCLUDE statements for the LNKDSL and DEPRTR (or RTRVSE on VSE) modules are included in the job.

This section covers the following topics:

Installing DBID/SVC Routing under z/OS Batch, TSO and IMS

The installation steps for the Adabas SVC routing feature under z/OS batch, TSO, and IMS are the same.

To install the Adabas DBID/SVC routing feature under z/OS batch, TSO, or IMS, complete the following steps:

1 Define or modify the DBID/SVC routing table by coding a series of MDBCSVC macros in a library member. Sample member ADASVCTB is provided in the ADA*vrs*.SRCE library as a template for preparing this member. For more information about using the MDBSVC macro, read ***Using the MDBSVC Macro***, elsewhere in this section.

2 Assemble and link-edit the DBID/SVC routing table member to create the table as a load module that you can make available to the application execution job step. The load module should be linked non-reusable and non-reentrant because the link routine subprogram LNKDSL will need to store the addresses of the Adabas SVC IDT headers in the DBID/SVC module to reduce the operating overhead on multiple commands accessing the same Adabas SVC.

3 Define or modify a link globals table for the execution environment. The following LGBLSET keywords are required to support the Adabas SVC routing feature:

LGBLSET Keyword Setting	Description
DYNDBSVC=YES	This keyword and setting indicate that Adabas SVC routing is active for this job step.
DBSVCTN=*name*	This keyword specifies the name of the DBID/SVC table for this job step. This name must match the name of the load module created to ensure the proper table is loaded when the link routine runs.

4 Assemble and link-edit the updated link globals table, as described for the appropriate TP monitor. For batch/TSO, read *Installing Adabas with Batch/TSO under Adabas 8*, in *Adabas Installation for z/OS*; for IMS, read *Installing Adabas with IMS TM under Adabas 8* , in *Adabas Installation for z/OS*.

5 Relink ADALNK or ADALNKR, making sure that the INCLUDE statements for the LNKDSL and DEPRTR modules are included in the job. Samples of the jobs used to relink ADALNK and ADALNKR are listed in the following table:

Link Routine	Sample Job		
	z/OS batch	TSO	IMS
ADALNK	LNKLNK8	LNKLNK8	---
ADALNKR	LNKLNKR8	LNKLNKR8	---
ADALNI8	---	---	LNKLNI8

Installing DBID/SVC Routing under z/VSE Batch

To install the Adabas DBID/SVC routing feature under z/VSE batch, complete the following steps:

1 Define or modify the DBID/SVC routing table by coding a series of MDBCSVC macros in a library member. Sample member ADASVCTB.A is provided in the sublibrary SAGLIB.ADA*vrs* as a template for preparing this member. For more information about using the MDBSVC macro, read ***Using the MDBSVC Macro***, elsewhere in this section.

2 Assemble and link-edit the DBID/SVC routing table member to create the table as a PHASE that you can make available to the application execution job step. The PHASE should be linked non-reusable and non-reentrant because the link routine subprogram LNKDSL will need to store the addresses of the Adabas SVC IDT headers in the DBID/SVC module to reduce the operating overhead on multiple commands accessing the same Adabas SVC.

3 Define or modify a link globals table for the execution environment. The following LGBLSET keywords are required to support the Adabas SVC routing feature:

LGBLSET Keyword Setting	Description
DYNDBSVC=YES	This keyword and setting indicate that Adabas SVC routing is active for this job step.
DBSVCTN=*name*	This keyword specifies the name of the DBID/SVC table for this job step. This name must match the name of the PHASE created to ensure the proper table is loaded when the link routine runs.

4 Assemble and link-edit the updated link globals table, as described for the appropriate TP monitor. For batch/TSO, read *Installing Adabas with Batch under Adabas 8*, in the *Adabas Installation for z/VSE*.

5 Relink ADALNK.PHASE or ADALNKR.PHASE, making sure that the INCLUDE statements for the LNKDSL and RTRVSE object modules are included in the job. Samples of the jobs used to relink ADALNK and ADALNKR are listed in the following table:

Link Routine	Sample Job
ADALNK.PHASE	ALNKLNK8.X
ADALNKR.PHASE	ALNKLNR8.X

Installing DBID/SVC Routing under CICS

To install the Adabas DBID/SVC routing feature under CICS, complete the following steps:

1 Define or modify the DBID/SVC routing table by coding a series of MDBCSVC macros in a library member. Sample member ADASVCTB is provided in the ADA*vrs*.SRCE library as a template for preparing this member. For more information about using the MDBSVC macro, read ***Using the MDBSVC Macro***, elsewhere in this section.

2 Assemble and link-edit the DBID/SVC routing table member to create the table as a load module and place it in a library that will be part of the CICS DFHRPL concatenation. The load module should be linked non-reusable and non-reentrant because the link routine sub-program LNKDSL will need to store the addresses of the Adabas SVC IDT headers in the DBID/SVC module to reduce the operating overhead on multiple commands accessing the same Adabas SVC.

3 Define the load module as a program to CICS using RDO, or the DFHCSDUP utility. See member DEFADA8 in the ACI*vrs*.SRCE libarary for sample DFHCSDUP definition statements. The program attributes should be Reload(No), Resident(Yes), Dataloc(Any), and Execkey(CICS).

4 Define or modify a link globals table for the execution environment. The following LGBLSET keywords are required to support the Adabas SVC routing feature:

LGBLSET Keyword Setting	Description
DYNDBSVC=YES	This keyword and setting indicate that Adabas SVC routing is active for this job step.
DBSVCTN=*name*	This keyword specifies the name of the DBID/SVC table for this job step. This name must match the name of the load module created to ensure the proper table is loaded when the link routine runs.

5 Assemble and link-edit the updated link globals table, as described in *Installing Adabas with CICS under Adabas 8* for z/OS installations or *Installing Adabas with CICS under Adabas 8* for z/VSE installations.

General Operation

When the Adabas SVC routing feature is installed, as described earlier in this section, it is loaded as described below:

- In batch, TSO, or IMS environments, the DBID/SVC routing table is loaded when the link routine initializes if the LGBLSET DYNDBSVC parameter is set to YES in the link globals table. The address of the routing table is kept in the link routine work area for use by all subsequent calls.
- In CICS environments, the Adabas 8 initialization module ADACIC0, normally run during PLTPI processing, loads and validates the DBID/SVC routing table, if the LGBLSET DYNDBSVC parameter was set to YES in the link globals table for the CICS region. The address of the routing table is kept in the global work area associated with the Adabas 8 task-related user exit (TRUE) module, ADACICT, and is made available on each application call to the TRUE by the Adabas command-level module ADACICS/ADADCI.

When an application call is made, the DBID/SVC routing table is searched by the LNKDSL subroutine which is linked with the appropriate link routine for each operating environment. LNKDSL is called after any LUEXIT1 (link routine user exit 1) is invoked, in case the pre-Adabas call user exit modifies the command's database ID for subsequent processing. The call to LNKDSL is made before any monitoring or Adabas Fastpath exits are called, so the monitoring product, such as Adabas Review, Adabas Fastpath, or Adabas Transaction Manager, will perform their processing based on the appropriate Adabas SVC found in the DBID/SVC routing table.

If the database ID associated with a particular call is not found in the DBID/SVC routing table, the default value for the Adabas SVC as specified by the MDBSVC macro's TYPE=INIT parameter is used. If the SVC located is not an Adabas SVC, or if it is not installed on the z/OS system, an Adabas response code of 213 with subcode 16 or 20 is returned to the application. If the calling database is not active for an SVC number, an Adabas response code of 148 is returned to the application.

Duplicate database IDs are not allowed in the DBID/SVC routing table as there is no reliable way for the link routine to determine which SVC should be used for a database ID if it is listed more than once. If duplicate database IDs are found while the table is being assembled, they are flagged with an assembler MNOTE and a return code of 16 is returned for the assembly attempt.

Using the MDBSVC Macro

Use the MDBSVC macro to define various aspects of the Adabas DBID/SVC routing table. Several MDBSVC macros are coded together using TYPE=INIT, TYPE=GEN, and TYPE=FINAL keywords to comprise a source module or member. This source module or member is then assembled and link-edited to build the DBID/SVC routing table load module. Sample member ADASVCTB in ADA*vrs*.SRCE can be used as a template for creating site-specific versions of the DBID/SVC routing table source module. Here is a sample DBID/SVC routing table source member that uses the CSECT name TESTDBT; when the table is assembled, its load module name will be TESTDBT:

```
TESTDBT CSECT
        MDBSVC TYPE=INIT,SVC=249,DBID=001
        MDBSVC TYPE=GEN,SVC=237,DBID=(2,10,21,33,175,1149),           X
               DBID2=(100,101,102,13500)
        MDBSVC TYPE=GEN,SVC=231,DBID=(226,899)
        MDBSVC TYPE=GEN,SVC=206,DBID=(15,16,69,99,500,12144)
        MDBSVC TYPE=GEN,SVC=248,DBID=(14,54,111,177,1213,5775)
        MDBSVC TYPE=GEN,SVC=249,DBID=(17,19,25,35,42,44,61,76)
        MDBSVC TYPE=FINAL
        END
```

When coding keyword values of MDBSVC macro statements, the assembler rules for continuing lines, identifying lists, and providing keyword values must be followed or assembly errors will result. Keywords and values with lists coded as objects of keywords must be separated by commas. There are no positional parameters used with the MDBSVC macro.

The MDBSVC macro can include the following four types of statements, as described in the following table:

MDBSVC Statement Type	Description	Number Allowed
TYPE=INIT	Only one MDBSVC TYPE=INIT statement can be included in the DBID/SVC routing table source member and it must be the first MDBSVC statement in the member. This statement identifies the beginning of the DBID/SVC routing table. The MDBSVC TYPE=INIT statement may also provide the default database ID and Adabas SVC number used for a call.	1
TYPE=GEN	Any number of MDBSVC TYPE=GEN statements can be included in the DBID/SVC routing table source member. These statements specify the lists of Adabas database IDs associated with specific valid Adabas SVC numbers.	any number, as needed.
TYPE=FINAL	Only one MDBSVC TYPE=FINAL statement can be included in the DBID/SVC routing table source member and it must be the last MDBSVC statement in the member before the assembler END statement. This statement identifies the end of the DBID/SVC routing table.	1
TYPE=DSECT	This statement type is reserved for Software AG internal use only. Do not use this statement type.	0

The MDBSVC TYPE=INIT statement can be preceded by a named CSECT statement and named AMODE and RMODE statements. If the CSECT, AMODE, or RMODE statements are included, the name used in them must agree with the name for the DBID/SVC routing table, as coded in the TABNAME parameter on the MDBSVC TYPE=INIT statement and as specified in the DBSVCTN keyword of the LGBLSET macro used for creating the link globals table.

This section covers the following topics:

- MDBSVC TYPE=INIT Syntax
- MDBSVC TYPE=GEN Syntax

MDBSVC TYPE=INIT Syntax

The syntax for the MDBSVC TYPE=INIT statement is:

```
MDBSVC TYPE=INIT [,SVC=svcno] [,DBID=dbid] [,TABNAME={name|ADBSVCT}] ↩
[,OPSYS={ZOS|VSE}]
```

The parameters you can code on the MDBSVC TYPE=INIT statement are described in ***MDBSVC Parameters***, elsewhere in this section.

MDBSVC TYPE=GEN Syntax

The syntax for the MDBSVC TYPE=GEN statement is:

```
MDBSVC TYPE=GEN [,SVC=svcno] [,DBID=id[, id]...][,DBID2=id[, id]...]
```

The parameters you can code on the MDBSVC TYPE=GEN statement are described in ***MDBSVC Parameters***, elsewhere in this section.

MDBSVC TYPE=FINAL Syntax

The syntax for the MDBSVC TYPE=FINAL statement is:

```
MDBSVC TYPE=FINAL
```

No parameters are valid on the MDBSVC TYPE=FINAL statement.

MDBSVC Parameters

The parameters that can be specified on various MDBSVC statements are as follows:

DBID

The DBID parameter can be coded on both the MDBSVC TYPE=INIT and MDBSVC TYPE=GEN statements.

- When specified on the MDBSVC TYPE=INIT statement, it lists the default database ID associated with the SVC specified in the SVC parameter. In this case, only one database ID can be listed in the DBID parameter on a TYPE=INIT statement.
- When specified on a MDBSVC TYPE=GEN statement, it lists the database IDs associated with the SVC specified in the SVC parameter. If more than one database ID is listed, they should be enclosed in parentheses and separated by commas.

Database IDs listed in the DBID parameter must be numeric and must correspond to the IDs of installed Adabas databases. In z/OS environments, database IDs must range from 1 to 65535. The same database ID cannot be specified on multiple MDBSVC statements; they must be

unique across all of the DBID and DBID2 statements in the DBID/SVC routing table. Duplicate values are flagged with an MNOTE, which causes the assembly of the DBID/SVC routing table to stop with return code 16.

The following is an example of some DBID parameters on various MDBSVC statements. Note that two MDBSVC statements list database IDs associated with SVC 237. This allows more database IDs to be coded for the same SVC number. Compare the way this is coded to the way the same example is coded for the DBID2 parameter. Both codings produce the same end result.

```
MDBSVC TYPE=INIT,SVC=249,DBID=1
MDBSVC TYPE=GEN,SVC=237,DBID=(2,4,10,16,21,33)
MDBSVC TYPE=GEN,SVC=237,DBID=(175,1149,1221)
MDBSVC TYPE=GEN,SVC=242,DBID=(3,18)
MDBSVC TYPE=FINAL
END
```

DBID2

The DBID2 parameter can be coded only on MDBSVC TYPE=GEN statements. It lists additional database IDs to be associated with an Adabas SVC specified in the SVC parameter. The DBID2 parameter is optional, but when it is specified, it must follow a DBID parameter.

Database IDs listed in the DBID2 parameter must be numeric and must correspond to the IDs of installed Adabas databases. In z/OS environments, database IDs must range from 1 to 65535. The same database ID cannot be specified on multiple MDBSVC statements; they must be unique across all of the DBID and DBID2 statements in the DBID/SVC routing table. Duplicate values are flagged with an MNOTE, which causes the assembly of the DBID/SVC routing table to stop with return code 16.

The following is an example of some MDBSVC statements that includes a DBID2 parameter. Compare the way this example is coded to the way the same example is coded for the DBID parameter. Both codings produce the same end result.

```
MDBSVC TYPE=INIT,SVC=249,DBID=1
MDBSVC TYPE=GEN,SVC=237,DBID=(2,4,10,16,21,33),                        X
       DBID2=(175,1149,1221)
MDBSVC TYPE=GEN,SVC=242,DBID=(3,18)
MDBSVC TYPE=FINAL
END
```

OPSYS

The OPSYS parameter is an optional parameter that can be coded only on the MDBSVC TYPE=INIT statement. This parameter identifies the operating system where the DBID/SVC routing table is assembled. Valid values for the OPSYS parameter are "ZOS" and "VSE"; the default is "ZOS".

PREFIX

The PREFIX parameter can only be coded only on the MDBSVC TYPE=DSECT statement, which is reserved for internal use by Software AG. Do not use this parameter.

SVC

The SVC parameter can be coded on both the MDBSVC TYPE=INIT and MDBSVC TYPE=GEN statements.

- When specified on the MDBSVC TYPE=INIT statement, it specifies the default Adabas SVC number to be used when the calling application provides a database ID that is not found in the DBID/SVC routing table.
- When specified on a MDBSVC TYPE=GEN statement, it specifies the Adabas SVC number to be associated with the Adabas databases identified by the DBID and DBID2 parameters.

The SVC number listed in the SVC parameter must be numeric and must correspond to the SVC number of an installed Adabas SVC. In z/OS environments, the SVC number must range from 200 to 255. Duplicate SVC values can be coded on multiple MDBSVC statements; this allows you to code long lists of database IDs and associate them with the same Adabas SVC.

In the following example, notice that there are two MDBSVC statements for SVC 249. It is the default SVC for the link routine and is also used for database 1, 3, and 18. There are also two MDBSVC statements for SVC 237; the two statements are used to list nine databases associated with SVC 237 (2, 4, 10, 16, 21, 33, 175, 1149, and 1221).

```
MDBSVC TYPE=INIT,SVC=249,DBID=1
MDBSVC TYPE=GEN,SVC=237,DBID=(2,4,10,16,21,33)
MDBSVC TYPE=GEN,SVC=237,DBID=(175,1149,1221)
MDBSVC TYPE=GEN,SVC=249,DBID=(3,18)
MDBSVC TYPE=FINAL
END
```

TABNAME

The TABNAME parameter is an optional parameter that can be coded only on the MDBSVC TYPE=INIT statement. This parameter specifies the name of the DBID/SVC routing table when the source member does not include a separate (and previously coded) CSECT statement. In this case, the name you specify on the TABNAME parameter is used to generate a named CSECT statement and named AMODE and RMODE directives.

The DBID/SVC routing table name that you specify should be between 1 and 8 alphanumeric characters long. In the following example, a DBID/SVC routing table with the name TESTDBT is coded.

```
MDBSVC TYPE=INIT,SVC=249,DBID=1,TABNAME=TESTDBT
MDBSVC TYPE=GEN,SVC=237,DBID=(2,4,10,16,21,33)
MDBSVC TYPE=GEN,SVC=237,DBID=(175,1149,1221)
MDBSVC TYPE=GEN,SVC=249,DBID=(3,18)
MDBSVC TYPE=FINAL
END
```

Modifying Source Member Defaults (LGBLSET Macro) in Version 8

The Adabas 8 LGBLSET macro is used to set default installation values for the Adabas link routines. It is used to prepare an object module which may either be link-edited with the Adabas 8 link routines or provided to the link routines in the job step where they are run. Your Adabas libraries include sample members provided to support the various teleprocessing (TP) monitors in each environment. Each of these sample members may be copied to an appropriate library and modified to provide the necessary customization required for the link routine that is intended to run in a given environment.

The LGBLSET parameter options with their default values (underlined) are described in the rest of this section:

- ADL: Adabas Bridge for DL/I Support
- AVB: Adabas Bridge for VSAM Support
- CITSNM: Adabas CICS TS Queue Name
- COR: SYSCOR Exit Support
- DBSVCTN: DBID/SVC Routing Table
- DYNDBSVC: DBID/SVC Routing Table
- ENTPT: Name of the Adabas CICS Command-Level Link Routine
- GBLNAME: Name of Link Globals Module
- GEN: Generate CSECT or DSECT
- IDTNAME: BS2000 IDT Common Memory Name
- IDTUGRP: BS2000 Memory Pool User Bound
- LOGID: Default Logical Database ID
- LUINFO: Length of User Data passed to Adabas LUEXIT1 and LUEXIT2
- LUSAVE: Size of User Save Area for Adabas LUEXIT1 and LUEXIT2
- LX1NAME: User Exit 1 Module Name
- LX2NAME: User Exit 2 Module Name
- MRO: Multiple Region Option
- NETOPT: Method Used to Create User ID
- NTGPID: Natural Group ID
- NUBS: Number of User Blocks Created By CICS Link Routine
- OPSYS: Operating System
- PARMTYP: Area for Adabas Parameter List
- PRE: DSECT Data Prefix
- PURGE: Purge Transaction
- RENT: Reentrant Module Flag
- RETRYX: Retry Command Exit Flag
- REVIEW: Adabas Review Support
- RMI: Resource Manager Interface
- RTXNAME: Command Retry Exit Name
- SAF: Adabas Security Interface Flag
- SAP: SAP Application Support

ADL: Adabas Bridge for DL/I Support

Parameter	Description	Syntax
ADL	Indicates whether or not the Consistency Interface of Software AG's Adabas Bridge for DL/I is to be supported by this command-level link routine. ■ ADL=YES: Adabas Bridge for DL/I Consistency Interface is to be supported. ■ ADL=NO: Adabas Bridge for DL/I Consistency Interface is *not* to be supported.	ADL={NO|YES}

AVB: Adabas Bridge for VSAM Support

Parameter	Description	Syntax
AVB	Indicates whether or not Software AG's Adabas Bridge for VSAM is to be supported by this command-level link routine. ■ AVB=YES: Adabas Bridge for VSAM is to be supported. ■ AVB=NO: Adabas Bridge for VSAM is *not* to be supported.	AVB={NO|YES}

CITSNM: Adabas CICS TS Queue Name

Parameter	Description	Syntax
CITSNM	Specifies the 16-byte string that represents the CICS TS queue name for Adabas. The default is "ADACICS".	CITSNM={ADACICS|*qname*}

COR: SYSCOR Exit Support

Parameter	Description	Syntax
COR	Indicates whether or not Adabas System Coordinator (SYSCOR), Adabas Transaction Manager, and Adabas Fastpath exits are installed and active. ■ COR=YES: The exits are installed and active. ■ COR=NO: The exits are *not* installed and active.	COR={NO|YES}

DBSVCTN: DBID/SVC Routing Table

Parameter	Description	Syntax
DBSVCTN	Provides the name of the DBID/SVC routing table that should be used by the link routine during its execution, if any. The routing table name must conform to names for z/OS standard load modules. It is used by a z/OS LOAD macro/SVC during batch, TSO, or IMS operation or by an EXEC CICS LOAD PROGRAM command during CICS operation. If the load module listed is not found, or if it is found to contain invalid header information, user abend U657 is issued in batch, TSO, or IMS environments. If the load module is not defined to CICS or not found in the CICS DFHRPL concatenation, the Adabas CICS link routine environment is not initialized. **Note:** If the DYNDBSVC parameter is set to NO, this parameter setting is ignored. For more information about SVC routing by database ID in z/OS environments, read *Establishing Adabas SVC Routing by Adabas Database ID*, in the *Adabas z/OS Installation GuideS* documentation. **Note:** Adabas client-based add-ons, such as Adabas Transaction Manager, are not compatible with this feature since for client-based functionality to work, it must be channeled through only a single router for any given session, not across routers. To avoid problems if the dynamic SVC by DBID routing feature in enabled for these products, error messages are issued, the assembly step of the globals table will receive return code 16, and the globals table load module will not be generated.	DBSVCTN={*name*|ADASVCTB}

DYNDBSVC: DBID/SVC Routing Table

Parameter	Description	Syntax
DYNDBSVC	Indicates whether Adabas SVC routing by database ID should be enabled for the link routine. DYNDBSVC=YES enables Adabas SVC routing by database ID; DYNDBSVC disables it. The default is NO. For more information about SVC routing by database ID in z/OS environments, read *Establishing Adabas SVC Routing by Adabas Database ID*, in the *Adabas z/OS Installation GuideS* documentation.	DYNDBSVC={YES|NO}

ENTPT: Name of the Adabas CICS Command-Level Link Routine

Parameter	Description	Syntax
ENTPT	The name given to the Adabas CICS command-level link routine. This name is used in EXEC CICS LINK commands to invoke Adabas services from CICS application programs. See also notes 1 and 2 in the installation procedure.	ENTPT={ADACICS|*name*}

GBLNAME: Name of Link Globals Module

Parameter	Description	Syntax
GBLNAME	The name of the link globals module.	GBLNAME={LNKGBLS|*name*}

GEN: Generate CSECT or DSECT

Parameter	Description	Syntax
GEN	Indicates whether a CSECT or DSECT is generated.	GEN={CSECT|DSECT}

IDTNAME: BS2000 IDT Common Memory Name

Parameter	Description	Syntax
IDTNAME	The common memory pool name of the BS2000 IDT.	IDTNAME=*name*

IDTUGRP: BS2000 Memory Pool User Bound

Parameter	Description	Syntax
IDTUGRP	Indicates whether the common memory pool is user bound (BS2000)	IDTUGRP={NO|YES}

LOGID: Default Logical Database ID

Parameter	Description	Syntax
LOGID	The value of the default target database ID. Valid ID numbers are 1-65535. The default is "1".	LOGID={*nnn* | 1}

LUINFO: Length of User Data passed to Adabas LUEXIT1 and LUEXIT2

Parameter	Description	Syntax
LUINFO	The length of the user data to be passed to target user exit 4. Valid values are numbers from zero (0) through 32,767. If LUINFO is not specified, the default is zero (no user data is passed).	LUINFO={0|*length*}

LUSAVE: Size of User Save Area for Adabas LUEXIT1 and LUEXIT2

Parameter	Description	Syntax
LUSAVE	The size of the user save area to be used by Adabas user exits LUEXIT1 and LUEXIT2. Valid values range from zero (0) through 256. The default is "72". If LUSAVE is not specified, the default is zero (no user save area is passed).	LUSAVE={72|*size*}

LX1NAME: User Exit 1 Module Name

Parameter	Description	Syntax
LX1NAME	The name of the link user exit 1 module	LX1NAME={LUEXIT1|*name*}

LX2NAME: User Exit 2 Module Name

Parameter	Description	Syntax
LX2NAME	The name of the link user exit 2 module	LX2NAME={LUEXIT2|*name*}

MRO: Multiple Region Option

Parameter	Description	Syntax
MRO	Indicates whether or not the CICS multiple region option (MRO) support is required. If you run the CICS command-level link with the CICS MRO, set this to MRO=YES; otherwise, use the default value MRO=NO. If MRO=YES, NETOPT must be set to NETOPT=NO (the default) to prevent non-unique LU names from multiple application regions. If NETOPT=YES and MRO=YES are specified, an assembler MNOTE and a return code of 16 are produced from the assembly step.	MRO={NO|YES}

NETOPT: Method Used to Create User ID

Parameter	Description	Syntax
NETOPT	If NETOPT=YES is specified, an 8-byte user ID will be constructed from the VTAM LU name. If NETOPT=NO is specified, the user ID is created from the constant CICS plus the four-byte CICS terminal ID (TCTTETI) for terminal tasks. For non-terminal tasks, the user ID comprises the constant CIC plus the CICS task number. If you run with the CICS multiple region option (MRO), you must use the default value for this option. If NETOPT=YES and MRO=YES are specified, an assembler MNOTE and a return code of 16 are produced from the assembly step.	NETOPT={NO|YES}

NTGPID: Natural Group ID

Parameter	Description	Syntax
NTGPID	Specifies a four-byte Natural group ID as required for unique Adabas user ID generation in the CICSplex environment with Natural Version 2.2.8 and above. The value is associated with all users who call the Adabas command-level link routine assembled with the specified value. There is no default value. If no value is specified, the Adabas internal user ID is built in the conventional manner.	NTGPID=*4-byte-value*

Parameter	Description	Syntax
	Any four-byte alphanumeric value may be specified, but it must be unique for each Adabas command-level link routine running in a CICSplex, or z/OS image. If more than one NTGPID is required (for example, both test and production Natural 2.2.8), more than one Adabas command-level link routine with associated TRUE must be generated. If you run with the CICS multiple region option (MRO), you may use NTGPID to provide a 4-byte literal for the Adabas communication ID to be used by the Adabas SVC when multiple application regions call Adabas.	

NUBS: Number of User Blocks Created By CICS Link Routine

Parameter	Description	Syntax
NUBS	The number of user blocks (UBs) to be created in the user block pool by the CICS link routine. The number of blocks must be large enough to handle the maximum possible number of concurrent Adabas requests. **Note:** The Adabas 6.2 and above command-level link routine obtains storage for the user blocks (the UB pool) above the 16-megabyte line.	NUBS={100|*blocks*}

OPSYS: Operating System

Parameter	Description	Syntax
OPSYS	The operating system in use.	OPSYS={ZOS|VSE|CMS|BS2}

PARMTYP: Area for Adabas Parameter List

Parameter	Description	Syntax
PARMTYP	The CICS area which is to contain the Adabas parameter list. "TWA" picks up the parameter list in the first six fullwords of the transaction work area (TWA). When PARMTYP=COM, the Adabas parameters are supplied in the CICS COMMAREA provided by the calling program with the EXEC CICS LINK command. The COMMAREA list for an ACB call must be at least 32 bytes long and begin with the label "ADABAS52". The COMMAREA list for an ACBX call must be at least 24 bytes long and begin with the label "ADABAS8X". In addition, the last ABD in the COMMAREA list for an ACBX call must be indicated by setting the VL-bit -- in other words, the high bit in the address must be on (X'80').	PARMTYP={ALL|COM ↩ |TWA}

Parameter	Description	Syntax
	PARMTYP=ALL (the default) uses both the COMMAREA and TWA to pass the Adabas parameters; in this case, the COMMAREA is checked first. We do not recommend that you attempt to map the CICS TWA to the Adabas 8 ACBX direct call. This is because the TWA is of finite size per transaction and because the TWA in not available at CICS startup. We therefore recommend that CICS programs using the Adabas 8 CICS link routines use the COMMAREA only for passing data.	

PRE: DSECT Data Prefix

Parameter	Description	Syntax
PRE	The two-byte string to be used as the DSECT data prefix. The default is "LG".	PRE={LG|*prefix*}

PURGE: Purge Transaction

Parameter	Description	Syntax
PURGE	The PURGE parameter is used when assembling with CICS 3.2 or above. If PURGE=YES is specified, the CICS WAIT EXTERNAL will contain PURGEABLE as one of its parameters, allowing the transaction to be purged by CICS if the DTIMOUT value is exceeded and PURGE is specified. If PURGE=NO (the default) is specified, the NONPURGEABLE option is generated.	PURGE={NO|YES}

RENT: Reentrant Module Flag

Parameter	Description	Syntax
RENT	Indicates whether the globals module is reentrant.	RENT={NO|YES}

RETRYX: Retry Command Exit Flag

Parameter	Description	Syntax
RETRYX	Indicates whether the retry command exit is active.	RETRYX={NO|YES}

REVIEW: Adabas Review Support

Parameter	Description	Syntax
REVIEW	Indicates whether or not Software AG's Adabas Review performance monitor is installed and active. When REVIEW=YES is specified, a work area of 512 bytes is set up for use by Adabas Review.	REVIEW={NO|YES}

RMI: Resource Manager Interface

Parameter	Description	Syntax
RMI	The RMI parameter is used to indicate whether or not the CICS Resource Manager Interface is in use. If RMI=YES is specified, the Adabas task-related user exit (TRUE) will be executed as a resource manager (RM) using the CICS Resource Manager Interface (RMI). RMI=YES is valid only when the Adabas Transaction Manager is installed, enabled, and available to users executing in the CICS environment. Consult the Adabas Transaction Manager documentation for additional instructions related to the installation of the Adabas TRUE.	RMI={NO|YES}

RTXNAME: Command Retry Exit Name

Parameter	Description	Syntax
RTXNAME	The name of the command retry exit module.	RTXNAME={LUEXRTR|*name*}

SAF: Adabas Security Interface Flag

Parameter	Description	Syntax
SAF	Indicates whether Software AG's Adabas SAF Security support is required.	SAF={NO|YES}

SAP: SAP Application Support

Parameter	Description	Syntax
SAP	Indicates whether or not SAP user ID generation is supported. If SAP=YES is specified, the program will detect a SAP initialization call and set the user ID for SAP applications from the constant provided on the initialization call, plus the field ACBADD2. For more information, refer to the supplementary information provided to customers using the SAP application system.	SAP={NO|YES}

SAPSTR: SAP ID String

Parameter	Description	Syntax
SAPSTR	The four-byte SAP ID string to use.	SAPSTR={'SAP*'|*string*}

SVCNO: Adabas SVC number

Parameter	Description	Syntax
SVCNO	The value of the Adabas SVC number. On z/OS systems, valid values range from 200-255 and the default is "249". On z/VSE systems, valid values range from 32-128 and the default is "45".	SVCNO=*nnn*

TPMON: Operating Environment

Parameter	Description	Syntax
TPMON	The TP monitor operating environment. Valid values should be specified as follows: ■ Specify "BAT" to use batch. ■ Specify "CICS" to use CICS. ■ Specify "COM" to use Com-plete. ■ Specify "IMS" to use IMS. ■ Specify "TSO" to use TSO. ■ Specify "UTM" to use UTM. **Caution:** Be sure to specify a TP monitor operating environment that is supported on the operating system you selected in the OPSYS parameter. In addition, if OPSYS=CMS is specified, the TPMON parameter should not be specified.	TPMON={BAT|CICS|COM|IMS}

TRUENM: CICS TRUE Name

Parameter	Description	Syntax
TRUENM	Specifies the module name of the Adabas CICS task-related user exit (TRUE). The default is ADACICT.	TRUENM={ADACICT|*name*}

UBPLOC: User Block Pool Allocation

Parameter	Description	Syntax
UBPLOC	Specifies whether the user block (UB) pool is to be obtained above (the default) or below the 16-megabyte line in CICS. The ECB used by the EXEC CICS WAIT WAITCICS or the EXEC CICS WAIT EXTERNAL is included in the UB pool. The UBPLOC=BELOW setting supports versions of CICS that do not allow ECBs above the 16-megabyte line; that is, CICS/ESA 3.2 or below. Refer to the IBM manual *CICS Application Programming Reference* for more information.	UBPLOC={ABOVE|BELOW}

UES: Universal Encoding Support

Parameter	Description	Syntax
UES	Indicates whether or not Universal Encoding Support (UES) is required.	UES={NO|YES}

USERX1: User Exit 1 Flag

Parameter	Description	Syntax
USERX1	Indicates whether or not user exit 1 is active.	USERX1={NO|YES}

USERX2: User Exit 2 Flag

Parameter	Description	Syntax
USERX2	Indicates whether or not user exit 2 is active.	USERX2={NO|YES}

XWAIT: XWAIT Setting for CICS

Parameter	Description	Syntax
XWAIT	Indicates whether a standard EXEC CICS WAITCICS (XWAIT=NO) or a WAIT EVENTS EXTERNAL (XWAIT=YES) will be executed by the Adabas 8 task-related user exit (TRUE). XWAIT=YES is the default. The CICS WAIT EVENTS EXTERNAL (XWAIT=YES) is the recommended interface for CICS/TS 1.1 and above. The CICS WAITCICS statement (XWAIT=NO) is provided for use with CICS/MVS 2.1.2 and for CICS/VSE 2.1 through 2.3. It may also be used for	XWAIT={NO|YES}

Parameter	Description	Syntax
	CICS/TS 1.1 and above, but may result in poor CICS transaction performance or unpredictable transaction results in busy CICS environments. **Note:** If XWAIT=NO is specified for use under CICS/ESA 3.3, IBM APAR PN39579 must be applied to the CICS/ESA 3.3 system. For CICS/TS 1.1 and above, this APAR is not required.	

Notes:

1. If XWAIT=NO is specified, the ADACICT (Adabas 8 TRUE) module issues an EXEC CICS WAIT-CICS command instead of the EXEC CICS WAIT EVENT command. XWAIT=YES conforms with recommended IBM usage of the WAIT and ECB lists in a high-transaction volume CICS system with CICS/TS Version 1.1 and above.
2. All EXEC CICS commands are processed by the CICS preprocessor; the LGBLSET parameters cause the subsequent assembly step to skip some of the statements.

XWAIT Posting Mechanisms

CICS WAITCICS (XWAIT=NO) can support a soft post of the specified ECB. This has the disadvantage of becoming a low priority dispatchable unit of work in a CICS environment, since the hand-postable work is not processed by CICS on every work cycle.

EXEC CICS WAIT EXTERNAL (XWAIT=YES), on the other hand, allows CICS to make use of its special post exit code, and will always be checked and processed (if posted) on every CICS work cycle.

For more details on the differences between the various CICS WAIT commands and their relationship to hard and soft posting mechanisms, consult the IBM *CICS Application Programming Reference* and the texts accompanying IBM APAR PN39579 or "Item RTA000043874" on the IBM InfoLink service.

XWAIT and the Adabas SVC / Router

The Adabas SVC is fully compatible with the XWAIT=YES setting. The SVC performs the necessary hard post for Adabas callers under CICS using the Adabas command-level link routine. The same SVC performs a soft post for batch callers where the hard post is not required.

5 Device and File Considerations

This section provides information for the following device- and system file-related topics:

- installing on fixed-block addressing (FBA) devices;
- defining new devices; and
- changing defaults for sequential files.

Supported z/VSE Device Types

The standard characteristics of the device types supported by Adabas are summarized in the following table. The Adabas block sizes and RABNs per track are provided for each component for each device type.

Device	Trks/Cyl	ASSO	DATA	WORK	PLOG/RLOG	CLOG	TEMP/SORT/DSIM	Notes
1512	7	1536:37	18944:37	18944:37	18944:37	18944:37	18944:37	
3375	12	2016:15	4092:8	4096:8	4096:8	4096:8	8608:4	
3380	15	2004:19	4820:9	5492:8	5492:8	4820:9	7476:6	2
3390	15	2544:18	5064:10	5724:9	5724:9	5064:10	8904:6	2
3512	16	4096:64	16384:16	16384:16	16384:16	16384:16	16384:16	
5121	15	2048:16	4096:8	4096:8	4096:8	4096:8	4096:8	
5122	15	4096:8	8192:4	8192:4	8192:4	8192:4	8192:4	
5123	15	4096:8	16384:2	16384:2	16384:2	16384:2	16384:2	
8345	15	4092:10	22780:2	22920:2	22920:2	22920:2	22920:2	
8380	15	3476:12	6356:7	9076:5	9076:5	9076:5	9076:5	1
8381	15	3476:12	9076:5	11476:4	11476:4	9076:5	9076:5	1
8385	15	4092:10	23292:2	23468:2	23468:2	23468:2	23468:2	1
8390	15	3440:14	6518:8	10706:5	10706:5	8904:6	8904:6	1
8391	15	4136:12	10796:5	13682:4	13682:4	8904:6	18452:3	1
8392	15	4092:12	12796:4	18452:3	18452:3	18452:3	18452:3	1
8393	15	4092:12	27644:2	27990:2	27990:2	27990:2	27990:2	1
9345	15	4092:10	7164:6	11148:4	11148:4	22920:2	22920:2	2

Notes:

1. The 8350, 838*n*, and 839*n* are pseudo-device types physically contained on a 3350, 3380, and 3390 device, respectively, but for which some or all of the standard block sizes are larger.
2. The IBM RAMAC 9394 emulates devices 3390 Model 3, 3380 Model K, or 9345 Model 2.

FBA Devices

Adabas Version 8.1 users must install three zaps before defining files on FBA devices. The zap numbers vary by 8.1 release and can be found in the Knowledge Center in Software AG's Empower (*https://empower.softwareag.com*) web site. For more information about applying zaps, read *Applying Zaps*, in the *Adabas Release Notes*.

- For Adabas Version 8.1.2, install zaps AD812064, AD812065, and AD812066.
- For Adabas Version 8.1.3, install zaps AD813034, AD813035, and AD813036.
- For Adabas Version 8.1.4, install zaps AD814004, AD814005, and AD814006.

All device definitions for Adabas control statements for FBA disks should specify one of the following devices types:

- FBA SCSI devices: Specify a device type of 1512.
- Virtual FBA devices: Specify device types of 5121, 5122, or 5123.

Note: Virtual FBA devices are not permanent and are, therefore, only suitable for holding temporary or work data sets.

Choose a device type based on the block sizes given in the following tables:

SCSI Device Types:

Dev Type	Asso blksz	Data blksz	Work blksz	Temp blksz	Sort blksz	PLOG blksz	CLOG blksz
1512	1536	18944	18944	18944	18944	18944	18944

Virtual FBA Device Types:

Dev Type	Asso blksz	Data blksz	Work blksz	Temp blksz	Sort blksz	PLOG blksz	CLOG blksz
5121	2048	4096	4096	4096	4096	4096	4096
5122	4096	8192	8192	8192	8192	8192	8192
5123	4096	16384	16384	16384	16384	16384	16384

The pseudo-cylinder for each of these devices has a different number of blocks as described below:

- 1512 cylinder = FBA blocks/777
- 5121 cylinder = FBA blocks/960
- 5122 cylinder = FBA blocks/960
- 5123 cylinder = FBA blocks/960

The size definitions for FBA devices on Adabas control statements can specify the number of pseudo-cylinders or the number of Adabas blocks (RABNs).

Make sure that the starting block and the number of FBA blocks on the z/VSE EXTENT statement are on an FBA pseudo-cylinder boundary, which is based on the device as specified above for each Adabas file comprising the database:

- An SCSI pseudo-cylinder (device type 1512) comprises 777 elements of 512 bytes each, or 388K per pseudo-cylinder. For example, an EXTENT entry for a ten cylinder SCSI device might consist of:

```
// EXTENT SYS123,,,,777,7770
```

- A virtual FBA pseudo-cylinder comprises 960 elements of 512 bytes each, or 480K per pseudo-cylinder. For example, an EXTENT entry for a ten cylinder virtual FBA device might consist of:

```
// EXTENT SYS123,,,,512,5120
```

ECKD Devices

Adabas supports ECKD DASD devices such as the IBM 3390 with the 3990 controller and ESCON channels.

During an open operation, ADAIOR determines which DASD device types are being used for the ASSO, DATA, WORK, SORT, and TEMP data sets. At that time, Adabas issues an informational message for each Adabas database component, where *type* is the component:

```
ADA164 ... FILE DDtype HAS BEEN OPENED IN ckd/eckd MODE - RABN SIZE rabn-size
```

Note: Software AG strongly recommends that you avoid mixing ECKD and CKD extents within a file, because the file will be opened only in CKD mode. Mixing extents could degrade performance when file I/O operations are performed.

Adding New Devices

Support for new device types that include user-defined block sizes can be implemented in ADAIOR by modifying one of the table of device-constant entries (TDCEs) reserved for this purpose.

A TDCE is X'40' bytes long and the first free TDCE can be identified by X'0000' in its first two bytes (TDCDT).

For Adabas Version 8.1, TDCE entries are in the ADAIOS CSECT TDCON, which corresponds to ESDID 1EC in object module IOSVSE.OBJ. The first TDCE entry is at offset X'19398' into IOSVSE.OBJ; the first free TDCE entry is at offset X'19898'.

This information is valuable when adding an additional TDCE entry, and when zapping the object module and relinking ADAIOS under z/VSE.

The z/VSE MSHP control statements to add a TDCE entry at the first free entry thus take the form:

```
// EXEC MSHP
CORRECT 9001-ADA-00-vrs :AD99998
AFFECTS MODULE=IOSVSE,ESDID=1EC
ALTER 19898 0000 : nnnn
ALTER 1989A 0000 : nnnn
.
. (etc.)
.
INVOLVES LINK=LNKIOS
/*
```


Information to be Zapped into the First Free ADAIOR TDCE

The information in the following tables must be zapped into the first free TDCE. The rules described in the section ***General Rules for Defining Device Block Sizes*** must be followed when changing the TDCE.

Label	Offset	Contents
TDCDT	00	Device type in unsigned decimal (X'3385'), must be numeric, and unique among all TDCEs.
TDCKSN	02	Constant set number: must be uniquely chosen from the values X'2B' or X'2E'.
TDCF	03	The flag bit must be set—TDCFCKD (X'40') for CKD devices, TDCFECKD (X'60') for ECKD devices or TDCFECKD (X'61') for ECKD, not user defined devices.
TDCDT1	04	(see note)
TDCDT2	05	(see note)
TDCDT3	06	(see note)
TDCDT4	07	(see note)
TDCMSBS	08	Refer to the TDCMSBS default table in *Maximum Sequential Block Size* in the Adabas z/OS installation instructions for more system- and device-related information.
TDCTPC	0A	Number of tracks per cylinder.
TDCCIPT	0C	Number of FBA blocks or PAM pages per track (if TDCFFBA is set).

Label	Offset	Contents
TDCBPCI	0E	Number of bytes per FBA block or PAM page (2048 if TDCFFBA is set).
TDCABPT	10	Number of Associator blocks per track.
TDCABS	12	Associator block size.
TDCACPB	14	Number of FBA blocks or PAM pages per Associator block (if TDCFFBA is set).
TDCDBPT	16	Number of Data Storage blocks per track.
TDCDBS	18	Data Storage block size.
TDCDCPB	1A	Number of FBA blocks or PAM pages per Data Storage block (if TDCFFBA is set).
TDCWBPT	1C	Number of Work blocks per track.
TDCWBS	1E	Work block size.
TDCWCPB	20	Number of FBA blocks or PAM pages per Work block (if TDCFFBA is set).
TDCTSBPT	22	Number of TEMP or SORT blocks per track
TDCTSBS	24	TEMP or SORT block size.
TDCTSCPB	26	Number of FBA blocks or PAM pages per TEMP or SORT block (if TDCFFBA is set).
TDCPBPT	28	Number of PLOG blocks per track.
TDCPBS	2A	PLOG block size.
TDCPCPB	2C	Number of FBA blocks or PAM pages per PLOG block (if TDCFFBA is set).
TDCCBPT	2E	Number of CLOG blocks per track.
TDCCBS	30	CLOG block size.
TDCCCPB	32	Number of FBA blocks or PAM pages per CLOG block (if TDCFFBA is set).

Note: One or more z/VSE codes for identifying the device type: PUB device type from PUBDEVTY (refer to the IBM MAPDEVTY macro).

General Rules for Defining Device Block Sizes

The following general rules must be followed when defining Adabas device block sizes:

- All block sizes must be multiples of 4.
- A single block cannot be split between tracks (that is, the block size must be less than or equal to the track size).

Block Rules for ASSO/DATA

The following rules are applicable for Associator and Data Storage:

- Associator block size must be greater than one-fourth the size of the largest FDT, and should be large enough to accept definitions in the various administrative blocks (RABN 1 - 30) and in the FCB;
- The block sizes for Associator and Data Storage should be a multiple of 256, less four bytes (for example, 1020) to save Adabas buffer pool space.

- The Associator and Data Storage block sizes must be at least 32 less than the sequential block size.
- Data Storage block size must be greater than: (maximum compressed record length + 10 + padding bytes).

Block Rule for WORK

The following rule is applicable for Work::

- The Work block size must be greater than either (maximum compressed record length + 110) or (Associator block size + 110), whichever is greater.

Block Rules for TEMP/SORT

The following rules are applicable for TEMP and SORT:

- Block sizes for TEMP and SORT must be greater than the block sizes for Data Storage.
- If ADAM direct addressing is used:

```
size > (maximum compressed record length + ADAM record length + 24);
size > 277 (maximum descriptor length + 24)
```

TEMP and SORT are generally read and written sequentially; therefore, the larger the TEMP/SORT block size, the better.

Block Rule for PLOG or SIBA

The following rules are applicable for PLOG and SIBA:

- The PLOG or SIBA block size must be greater than either (maximum compressed record length + 110) or (Associator block size + 110), whichever is greater.
- It is also recommended that PLOG/SIBA be defined larger than the largest Data Storage block size. This avoids increased I/O caused by splitting Data Storage blocks during online ADASAV operations.

  The block size (BLKSIZE) of a sequential file is determined as follows:

```
if PTTF(JCL) then BLKSIZE is taken from file assignment statement or label;
if PTTMBS > 0 then BLKSIZE = PTTMBS;
if PTTMBS = 0 then
if tape then BLKSIZE = 32760;
else BLKSIZE = TDCMSBS;
else if BLKSIZE in file assignment statement or label then use it;
if PTTF(OUT) then
if QBLKSIZE > 0 then BLKSIZE = QBLKSIZE;
```

```
if tape then BLKSIZE = 32760;
else BLKSIZE = TDCMSBS;
else error.
```

Note: `QBLKSIZE` is an ADARUN parameter.

Using 3480/3490 Tape Cartridge Compression (IDRC)

The use of hardware compression (IDRC) is not recommended for protection log files. The ADARES BACKOUT function will run much longer when processing compressed data. Also, the BACKOUT function is not supported for compressed data.

User ZAPs to Change Logical Units

The specified zaps should be added to the module IOSVSE / phase ADAIOS, not to the specified utility.

PTT entries are in the ADAIOS CSECT I_PTT. The first PTT entry is at offset 0 into CSECT I_PTT.

When zapping the object module and relinking ADAIOS, note that the ADAIOS CSECT I_PTT corresponds to ESDID 000D in object module IOSVSE.OBJ. The offset of CSECT I_PTT into IOSVSE.OBJ is X'1000'.

Utility	File	Default SYS Number	PTT Offset	VER	REP
ADACDC	SIIN	SYS010	6B8	1A0A	1Axx
ADACMP	AUSBA	SYS012	08	820C	82xx
	AUSB1	SYS021	18	8215	82xx
	AUSB2	SYS022	28	8216	82xx
	AUSB3	SYS023	38	8217	82xx
	AUSB4	SYS024	48	8218	82xx
	AUSB5	SYS025	58	8219	82xx
	AUSB6	SYS026	68	821A	82xx
	AUSB7	SYS027	78	821B	82xx
	AUSB8	SYS028	88	821C	82xx
	AUSB9	SYS029	98	821D	82xx
	EBAND	SYS010	A8	180A	18xx
	FEHL	SYS014	B8	820E	82xx
ADACNV	FILEA (OUTPUT)	SYS010	698	820A	82xx
	FILEA (INPUT)	SYS010	6A8	120A	12xx

Utility	File	Default SYS Number	PTT Offset	VER	REP
ADALOD	FILEA (OUTPUT)	SYS012	D8	820C	82xx
	FILEA (INPUT)	SYS012	E8	020C	02xx
	EBAND	SYS010	F8	1A0A	1Axx
	ISN	SYS016	108	1A10	1Axx
	LOB (OUTPUT)	SYS017	118	8211	82xx
	LOB (INPUT)	SYS017	128	1A11	1Axx
	OLD	SYS014	138	820E	82xx
ADAMER	EBAND	SYS010	148	1A0A	1Axx
ADANUC	LOG	SYS012	158	820C	82xx
	SIBA	SYS014	168	C20E	C2xx
ADAORD	FILEA (OUTPUT)	SYS010	178	820A	82xx
	FILEA (INPUT)	SYS010	188	120A	12xx
ADAPLP	PLOG	SYS014	198	1A0E	1Axx
ADARAI	OUT	SYS010	6C8	800A	80xx
ADAREP	SAVE	SYS010	1A8	1A0A	1Axx
	PLOG	SYS011	1B8	1A0B	1Axx
ADARES	SIIN	SYS020	1C8	1A14	1Axx
	BACK	SYS020	1D8	2C14	2Cxx
	SIAUS1	SYS021	1E8	8215	82xx
	SIAUS2	SYS022	1F8	8216	82xx
ADASAV	SAVE1	SYS011	208	820B	82xx
	SAVE2	SYS012	218	820C	82xx
	SAVE3	SYS013	228	820D	82xx
	SAVE4	SYS014	238	820E	82xx
	SAVE5	SYS015	248	820F	82xx
	SAVE6	SYS016	258	8210	82xx
	SAVE7	SYS017	268	8211	82xx
	SAVE8	SYS018	278	8212	82xx
	DUAL1	SYS021	288	8215	82xx
	DUAL2	SYS022	298	8216	82xx
	DUAL3	SYS023	2A8	8217	82xx
	DUAL4	SYS024	2B8	8218	82xx
	DUAL5	SYS025	2C8	8219	82xx
	DUAL6	SYS026	2D8	821A	82xx
	DUAL7	SYS027	2E8	821B	82xx
	DUAL8	SYS028	2F8	821C	82xx

Utility	File	Default SYS Number	PTT Offset	VER	REP
	REST1	SYS011	308	1A0B	1Axx
	REST2	SYS012	318	120C	12xx
	REST3	SYS013	328	120D	12xx
	REST4	SYS014	338	120E	12xx
	REST5	SYS015	348	120F	12xx
	REST6	SYS016	358	1210	12xx
	REST7	SYS017	368	1211	12xx
	REST8	SYS018	378	1212	12xx
	FULL	SYS030	388	1A1E	1Axx
	DEL1	SYS031	398	1A1F	1Axx
	DEL2	SYS032	3A8	1A20	1Axx
	DEL3	SYS033	3B8	1A21	1Axx
	DEL4	SYS034	3C8	1A22	1Axx
	DEL5	SYS035	3D8	1A23	1Axx
	DEL6	SYS036	3E8	1A24	1Axx
	DEL7	SYS037	3F8	1A25	1Axx
	DEL8	SYS038	408	1A26	1Axx
	PLOG	SYS010	418	1A0A	1Axx
ADASEL	EXPA1	SYS011	428	820B	82xx
	EXPA2	SYS012	438	820C	82xx
	EXPA3	SYS013	448	820D	82xx
	EXPA4	SYS014	458	820E	82xx
	EXPA5	SYS015	468	820F	82xx
	EXPA6	SYS016	478	8210	82xx
	EXPA7	SYS017	488	8211	82xx
	EXPA8	SYS018	498	8212	82xx
	EXPA9	SYS019	4A8	8213	82xx
	EXPA10	SYS020	4B8	8214	82xx
	EXPA11	SYS021	4C8	8215	82xx
	EXPA12	SYS022	4D8	8216	82xx
	EXPA13	SYS023	4E8	8217	82xx
	EXPA14	SYS024	4F8	8218	82xx
	EXPA15	SYS025	508	8219	82xx
	EXPA16	SYS026	518	821A	82xx
	EXPA17	SYS027	528	821B	82xx
	EXPA18	SYS028	538	821C	82xx

Utility	File	Default SYS Number	PTT Offset	VER	REP
	EXPA19	SYS029	548	821D	82xx
	EXPA20	SYS030	558	821E	82xx
	SIIN	SYS010	568	1A0A	1Axx
ADATRA	TRA	SYS019	578	820A	82xx
ADAULD	OUT1	SYS010	588	820A	820xx
	OUT2	SYS011	598	820B	820xx
	ISN	SYS012	5A8	820C	820xx
	SAVE	SYS013	5B8	1A0D	1Axx
	PLOG	SYS014	5C8	1A0E	1Axx
	FULL	SYS030	5D8	1A1E	1Axx
	DEL1	SYS031	5E8	1A1F	1Axx
	DEL2	SYS032	5F8	1A20	1Axx
	DEL3	SYS033	608	1A21	1Axx
	DEL4	SYS034	618	1A22	1Axx
	DEL5	SYS035	628	1A23	1Axx
	DEL6	SYS036	638	1A24	1Axx
	DEL7	SYS037	648	1A25	1Axx
	DEL8	SYS038	658	1A26	1Axx
ADAVAL	FEHL	SYS014	668	820E	820xx

6 Enabling Universal Encoding Support (UES) for Your Adabas Nucleus

Prior to Adabas Version 7, Entire Net-Work converted all data for mainframe Adabas when necessary from ASCII to EBCDIC. Starting with Version 7, Adabas is delivered with its own data conversion capability called *universal encoding support (UES)*. Entire Net-Work detects when it is connected to a target database that converts data and passes the data through to Adabas without converting it.

Universal encoding support must be activated in:

- The Adabas nucleus. The steps to implement UES in the Adabas nucleus are provided in ***Installing UES Support for the Adabas Nucleus***, elsewhere in this chapter.
- The Adabas link routines. For Adabas Version 7, UES is enabled by default for the link routines ADALNK, ADALNKR, and ADALCO. For Adabas 8, UES is enabled by default for *all* link routines. For information on altering UES enablement in the link routines read appropriate sections of ***Installing Adabas With TP Monitors***, elsewhere in this guide, starting with the section ***UES-Enabled Link Routines***.

  UES-enabled databases can be connected to machines with different architectures through Com-plete, Software AG internal product software (APS), or through Entire Net-Work (WCP). Connections through Com-plete or Software AG internal product software (APS) use the Adabas Com-plete link routines; connections through Entire Net-Work use the Adabas batch link routines.
- The Adabas database. For more information, read *Universal Encoding Support (UES)* in *Adabas DBA Tasks Manual* as well as *ADADEF Utility: Define a Database* and *ADACMP Utility: Compress-Decompress Data* in *Adabas Utilities Manual* for more information.

Note: The use of UES-enabled link routines and a UES-enabled nucleus is transparent to applications, including applications that do not require universal encoding translation support. Therefore, it is not necessary to disable UES if it is already enabled.

Installing UES Support for the Adabas Nucleus

The following LIBR sublibraries are distributed with ADABAS for UES support:

```
SAGLIB.ADAvrsCS

SAGLIB.APS272nn
SAGLIB.APS272
```

To install these libraries:

1 Create a z/VSE sublibrary for the code pages.

```
* $$ JOB JNM=CRUESL,CLASS=0,DISP=D,LDEST=(,xxxxxx)
* $$ LST CLASS=A,DISP=D
// JOB LIBRDEF
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,'SAG.ADABAS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// EXEC LIBR
DEFINE L=DDECSOJ R=Y
/*
/&
* $$ EOJ
```

2 Create a z/VSE sublibrary for APS.

```
* $$ JOB JNM=CRUAPS,CLASS=0,DISP=D,LDEST=(,xxxxxx)
* $$ LST CLASS=A,DISP=D
// JOB LIBRDEF
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,'SAG.APS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// EXEC LIBR
DEFINE L=APS272nn R=Y
DEFINE L=APS272      R=Y
/*
/&
* $$ EOJ
```

3 Restore the UES code pages sublibrary to this file. Refer to the *Report of Tape Creation* for the file positions on the distribution tape.

```
* $$ JOB JNM=RESTECS,DISP=D,CLASS=0,LDEST=(,xxxxxx)
* $$ LST DISP=D,CLASS=A
// JOB RESTECS
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL ADALIB,'SAG.ADABAS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// ASSGN SYS006,cuu
// MTC REW,SYS006
// EXEC LIBR,PARM='MSHP'
RESTORE SUBLIB=SAGLIB.ADAvrsCS:ADALIB.DDECSOJ -
TAPE=SYS006 -
LIST=YES -
REPLACE=YES
/*
// MTC REW,SYS006
// ASSGN SYS006,UA
/&
* $$ EOJ
```

4 Restore the APS sublibraries. Refer to the *Report of Tape Creation* for the file positions on the distribution tape.

```
* $$ JOB JNM=RESTAPS,DISP=D,CLASS=0,LDEST=(,xxxxxx)
         * $$ LST DISP=D,CLASS=A
         // JOB RESTAPS
         // ASSGN SYS005,DISK,VOL=vvvvvv,SHR
         // DLBL APSLIB,'SAG.APS.LIB',2099/365,SD
         // EXTENT SYS005,vvvvvv,1,0,ssss,tttt
         // ASSGN SYS006,cuu
         // MTC REW,SYS006
         // EXEC LIBR,PARM='MSHP'
         RESTORE SUBLIB=SAGLIB.APS272nn:ADALIB.APS272nn -
         TAPE=SYS006 -
         LIST=YES -
         REPLACE=YES
          /*
         // MTC REW,SYS006
         // ASSGN SYS006,UA
         /&
         * $$ EOJ
```

5 Repeat the previous step for SAGLIB.APS272.

6 Modify the Adabas startup JCL, adding the UES environment section after the ADARUN parameters:

```
ADARUN .....
ADARUN ....
/*
ENVIRONMENT_VARIABLES=/DDECSOJ/ADAvrs/ENVVARS.P
/*
/&
* $$ EOJ
```

Reference the library where the libraries were restored in your Adabas startup procedure:

```
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,SAG.ADABAS.LIB,2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
```

And add the libraries to the LIBDEF chain:(be sure SAGLIB.APS272*nn* is referenced *before* SAGLIB.APS272):

```
// DLBL APSLIB,SAG.APS.LIB,2099/365,SD
// EXTENT SYSOO6,vvvvvv,1,0,ssss,tttt
// LIBDEF PHASE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs,...X
DDECSOJ.DDECSOJ, X
APSLIB.APS272nn,APSLIB.APS272)
// LIBDEF OBJ,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
DDECSOJ.APS272nn,DDECSOJ.APS272)
// LIBDEF SOURCE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs,... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
APSLIB.APS272nn,APSLIB.APS272)
// LIBDEF PHASE,CATALOG=SAGLIB.USRLIB
```

7 Modify the ENVVARS.P file, adding the following line in the APS*vrs* library:

```

* This member contains Environment Variables used by APS and
* APS-based applications.
*
ECSOBJDIR=FILE://DDECSOJ/DDECSOJ
```

8 Run the ADADEF utility setting `UES=YES`:

```
* $$ JOB JNM=ADADEF,CLASS=0,DISP=D,LDEST=(,xxxxxx)
* $$ LST CLASS=A,DISP=D
// JOB ADADEF EXECUTE THE ADABAS VERSION 7 UTILITY ***DEF***
// OPTION LOG,PARTDUMP
*
// EXEC PROC=ADALIB
*
// EXEC PROC=ADAFIL
*
// EXEC ADARUN,SIZE=ADARUN
*
ADARUN PROG=ADADEF,MODE=SINGLE,SVC=svc,DEVICE=dddd,DBID=nnnn
/*
ADADEF MODIFY UES=YES
/*
/&
* $$ EOJ
```

9 Start the database.

You should see the following message:

```
ENTIRE CONVERSION SERVICES INITIALIZED
```

7 Installing The AOS Demo Version

This section describes how to install the Adabas Online System (AOS) demo version. To install AOS on systems that use Software AG's System Maintenance Aid (SMA), refer to the section of this document describing installation of Adabas in your operating environment. For information about SMA, see the *System Maintenance Aid* documentation.

The AOS demo version requires Natural version 3.1 or above.

Note: To install the full version of Adabas Online System (AOS), read the *Adabas Online System* documentation.

AOS Demo Installation Procedure

To install the AOS demo version without the System Maintenance Aid

1 For a Com-plete or CICS environment, link the correct object module with the Natural TP nucleus.

If a split Natural nucleus is to be installed, the AOSASM module must be linked to the shared portion of the nucleus and not to the thread portion.

2 Perform a Natural INPL.

The tape containing the AOS demo version contains an INPL-formatted data set in Natural 3.1. The programs for the AOS demo version are stored in library SYSAOS.

The distributed INPL jobs (both the sample jobs and the SMA-generated jobs) that you use to load the Adabas INPL library load it in a date-sensitive manner. In other words, the load process will now check the dates of your existing INPL library and will not allow older members to overwrite members with newer dates. However, if you use your own Natural batch jobs to load the Adabas INPL library, you will need to modify them to be date-sensitive. To do this, specify the following `CMSYNIN` primary command input in your job (this setting assumes the Natural input parameters in the job are specified in comma-delimited mode, or `IM=D`):

```
B,,,,,,,Y
```

The "B" setting indicates that the INPL action should load everything; the next six fields (comma-delimited) are defaults, the eighth field is specified as "Y" to indicate that dates in the INPL library should be checked, and the ninth field is not included in the specification because the default for that field will be used. For more information about Natural `CMSYNIN` input, refer to your Natural documentation.

Note: When migrating an Adabas 7.4 installation, this procedure does not apply. Instead, you should replace the 7.4 INPL library members with the latest Adabas 8 INPL library

members, regardless of the dates of the members, to avoid creating a library containing members from both releases.

3 Load the ADA error messages using the Natural utility ERRLODUS.

The error messages are stored in an ERRN-formatted data set included on the tape.

See the *Natural Utilities* documentation for information about the ERRLODUS utility.

4 Execute the AOS demo version by logging on to the application library SYSAOS and entering the command `MENU`.

Installing AOS with Natural Security

Natural Security must be installed before implementing Adabas Online System Security. See the *Adabas Security Manual* for more information. For information about installing Natural Security for use with AOS Security, see the *Natural Security Manual.*

Natural Security Version 3.1 or above includes the ability to automatically close all open databases when the Natural command mode's LOGON function of the AOS demo version is invoked.

Use the following procedure if Natural Security is installed in your environment.

1 Define at least the library SYSAOS to Natural Security

Software AG recommends you define this library and any others you may define as protected.

2 Specify the startup program for SYSAOS as MENU

Do not specify a startup program name for the other libraries.

Setting the AOS Demo Version Defaults

Parameters that control the operation of the AOS demo version can be set at installation time by changing the defaults in the Natural program AOSEX1. The table below lists the parameters and possible values. Default values are underlined:

Parameter	Valid Values / Default	Function
AOS-END-MSG	Yes (Y) / No (N)	Display the AOS demo version end-of-session message?
AOS-LOGO	Yes (Y) / No (N)	Display the AOS demo version logo?
CPEXLIST	No (N): normal list Yes (Y): extended	Display extended checkpoint list?
MAX-AC-IOS	0-999999 (150)	AC read converter block threshold value
NR-EXT	1, 2, 3, 4, 5	Critical extent threshold for listing file
STATINTV	1-9999 seconds (60)	Statistics gathering interval

To change the defaults, you must edit the Natural AOSEX1 program and make the changes directly within the program listing in the defaults area, as shown by the following example:

```
DEFINE DATA PARAMETER USING ADVPUEX1
END-DEFINE
*
* SET THE DEFAULTS
*
AOS-END-MSG = 'Y' (Display end-of-session message)
AOS-LOGO = 'Y' (Online System logo display—set to 'N' for no logo display)
CPEXLIST = 'N' (Checkpoint list control: set to 'Y' for extended checkpoint list)
NR-EXT = 4 (Critical extent threshold: 1, 2, 3, 4 or 5)
MAX-AC-IOS = 150 (AC read converter block threshold)
STATINTV = 60 (Statistic gathering time interval:  range: 1 - 9999)
*
END
```

8 Installing The Recovery Aid (ADARAI)

This section describes how to install the Adabas Recovery Aid (ADARAI).

ADARAI Installation Overview

To install the Adabas Recovery Aid, it is necessary to:

- allocate the recovery log;
- customize the skeleton job streams for your installation (see the *Adabas Operations* documentation for more detailed information);
- update the necessary nucleus run/utility job control to include the Recovery Aid data definition statements;
- install the Adabas/ADARAI utility configuration; and
- run ADARAI PREPARE and a save operation to begin a logging generation.

ADARAI Installation Procedure

Except for customizing the skeleton job stream, the specific installation steps are as follows:

To install the Adabas Recovery Aid:

1 Allocate the recovery logs

Define and format the RLOGR1 file.

Use the ADAFRM RLOGFRM function to format the RLOGs.

2 Add data definition statements

Add an RLOGR1 DLBL statement to the nucleus job stream and to any utilities that update or save the database and thus write to the RLOG files.

Whenever these utilities are executed while ADARAI is active in the database (that is, after the PREPARE function has been executed), the RLOGR1 DLBL statement must be included.

The following utilities update the database and therefore write to the RLOG:

```
ADAORD (all STORE and REORDER functions)
ADALOD (all functions)
ADAINV (all functions)
ADARES REGENERATE/BACKOUT database
ADASAV RESTORE (all functions) and RESTPLOG
ADADEF NEWWORK
```

The following utilities save the database and therefore write to the RLOG:

```
ADASAV SAVE (all functions)
ADAORD RESTRUCTURE
ADAULD
```

The following utility functions have an impact on recovery and therefore write to the RLOG:

```
ADARES PLCOPY/COPY
ADASAV MERGE
```

Additionally, the Adabas nucleus writes to the RLOG during startup and termination. The nucleus also writes checkpoint information to the RLOG when ADADBS or Adabas Online System functions are processed, ensuring these events are known to ADARAI for recovery processing.

3 Install ADARAI on the database.

Execute the ADARAI PREPARE function. ADARAI PREPARE updates the ASSO GCBs to indicate that ADARAI is installed. It also creates a control record on the RLOG file with necessary ADARAI information (number of generations, RLOG size, etc.).

4 Create the first ADARAI generation.

Execute ADASAV SAVE (database) to start the logging of RLOG information. See the *Adabas Utilities* documentation for more information.

Once ADARAI is active in the database, protection logging must always be used.

9 Adabas Dump Formatting Tool (ADAFDP)

This section describes the use of the Adabas dump formatting tool ADAFDP.

ADAFDP Function

ADAFDP is the address space dump formatting module. During abnormal shutdown of the Adabas nucleus, this module receives control to format and display information that should help you analyze the reason for the error.

During a nucleus shutdown, ADAMPM determines the shutdown reason. If the reason is abnormal termination, ADAMPM loads the ADAFDP module into the address space prior to the 20 call to the Adabas SVC. ADAFDP subsequently receives control to format nucleus information.

If ADAFDP cannot be loaded, message ADAF03 is written to the console and abnormal shutdown continues.

ADAFDP Output

Much of the information formatted by ADAFDP is self-explanatory. However, because the type and amount of information depends on the shutdown situation, a summary of ADAFDP output is provided in this section.

- ADAFDP Messages
- Pool Abbreviations
- User Threads
- Command Information
- RABN Information

ADAFDP Messages

Message	Description
ADAH51 / ADAH52	The message is displayed on the console and written to DDPRINT at the point where the format begins and terminates.
ADAMPM ABEND CODE and PSW	If an Abend code and program status word (PSW) were saved in ADAMPM by the Adabas ESTAE, ADAFDP displays these. In addition, ADAFDP determines the module whose entry point best fits the PSW and calculates the offset within that module. If the ADAMPM abend code and PSW are zero, ADAFDP does not format this information.
ADABAS MODULE LOCATIONS	ADAFDP formats and displays the location of each of the Adabas nucleus modules resident in the address space.
ADDRESS LOCATIONS FOR USER EXITS	ADAFDP formats and displays the location of any user exit loaded with the Adabas nucleus.

Message	Description
ADDRESS LOCATIONS FOR HYPEREXITS	ADAFDP formats and displays the location of any hyperexit loaded with the Adabas nucleus. Hyperexits 10-31 are displayed as A-U, respectively.
ADANC0 STANDARD REGISTER SAVE AREA	Registers 0-7/8-F, which are saved in ADANC0. ADAFDP determines if any of these registers contains an address that points at a nucleus pool in storage. If yes, ADAFDP indicates which pool and snaps storage at that address. If the register is 12 and it points to a user thread, ADAFDP snaps the entire thread.
ADANC0 ABEND SAVE REGISTERS	Registers 0-7/8-F, which are saved in ADANC0 as a result of a user abend. ADAFDP determines if any of these saved registers contains an address that points at a nucleus pool in storage. If yes, ADAFDP indicates which pool and snaps storage at that location. If the saved register is 12 and it points to a user thread, ADAFDP snaps the entire thread.
ADAMPM SAVE REGISTERS	Registers 0-7/8-F, which were saved in ADAMPM by the Adabas ESTAE. These are the same registers displayed with the ADAM99 message. ADAFDP determines if any of these saved registers contains an address that points within a nucleus pool in storage. If yes, ADAFDP indicates which pool and snaps storage at that location.
BEGIN / ENDING ADDRESSES OF POOLS / TABLES	ADAFDP determines begin/ending address locations for pools and tables for the Adabas nucleus. These addresses are presented for easy location in the actual dump. See ***Pool Abbreviations*** for more information.
ADABAS THREADS	ADAFDP formats the physical threads including threads 0, -1, and -2. The number of lines depends on the value of `NT`. The thread that was active at the time of the abnormal termination (if any) is marked by a pointer "->".
USER THREADS	For any of the threads -2 to `NT` that had assigned work to perform, ADAFDP formats and displays information about the status of that thread. See ***User Threads*** for more information:
FOLLOWING COMMANDS WERE FOUND IN THE CMD QUEUE	ADAFDP scans the command queue and formats information for any command found in the queue. See ***Command Information*** for more information.
POOL INTEGRITY CHECK	ADAFDP check the integrity of several pools within the Adabas nucleus address space. If an error is detected within that pool, ADAFDP indicates which pool and what type of error was encountered. In addition, ADAFDP snaps storage at the location where the error was detected.
FOLLOWING RABNS / FILES ACTIVE IN BUFFER POOL	ADAFDP scans the buffer pool header for RABNs that were active or being updated. See ***RABN Information*** for more information.
ADAIOR REGS FOUND AT OFFSET X'080'	Registers 0-7/8-F found saved in ADAIOR at this offset. If ADAFDP determines that any of these register values is pointing within an Adabas pool, it snaps storage at that location.
ADAIOR REGS FOUND AT OFFSET X'0C0'	Registers 0-7/8-F found saved in ADAIOR at this offset. If ADAFDP determines that any of these register values is pointing within an Adabas pool, it snaps storage at that location.
ICCB POINTED FROM X'A0' IN IOR	The ICCB address to which this offset in ADAIOR points.

Message	Description
ADAI22 ADAIOR TRACE TABLE	Format of ADAIOR trace table; same as that found with the ADAM99 message.

Pool Abbreviations

Pool Abbreviation	Description
LOG	Log area
OPR	Adabas nucleus operator command processing area
CQ	Address of the command queue, which is formatted later by ADAFDP
ICQ	Internal command queue
TT	Thread table
IA1	Software AG internal area 1
SFT	Session file table
FU	File usage table
FUP	File update table
IOT	I/O table for asynchronous buffer flushing
PL2	PLOG area for asynchronous buffer flushing
PET	Table of posted ETs
TPT	Tpost
TPL	Tplatz
UQP	Unique descriptor pool
UHQ	Upper hold queue
HQ	Hold queue
UUQ	Upper user queue
UQ	User queue
FP	Format pool
FHF	File HILF element
PA	Protection area
TBI	Table of ISNs
TBQ	Table of sequential searches
WK3	Work part 3 space allocation table
IA2	Software AG internal area 2
WK2	Work part 2 space allocation table
VOL	VOLSER table
WIO	Work block I/O area
FST	Free space table work area

Pool Abbreviation	Description
UT	User threads
WP	Work pool
AW2	Work block asynchronous I/O area
IOP	I/O pool related to asynchronous buffer flush
IU2	Buffer pool importance header upper 2
IU1	Buffer pool importance header upper 1
BU2	Buffer pool upper header 2
BU1	Buffer pool upper header 1
BH	Address location of the buffer pool header, information from the buffer pool header is formatted later by ADAFDP
BP	Address location of the physical start of the buffer pool

User Threads

Information	Description
Thread Number	-2 to NT
Status	Indicates the current status of the thread: ■ *Active*: the currently active thread ■ In Use: thread has been assigned work ■ Waiting For I/O: waiting for a block not in buffer pool ■ Waiting For RABN: waiting for a RABN already in use ■ Waiting For Work-2 Area Block: similar to waiting for I/O ■ Waiting Workpool Space: provides number of bytes in decimal ■ Ready To Run: waiting to be selected for execution
CMD	The Adabas command being executed
Response Code	Response code (if any)
File Number	File number for this command
ISN	Internal sequence number for this command
Sub. Rsp	Subroutine response code (if any)
Last RABN for I/O	Last RABN required by command processing, in decimal
Type	Last RABN type (A - ASSO, D - DATA)
CQE Addr	Command queue element address for this command
User Jobname	Job name for user who executed this command
ITID	Internal Adabas ID for user who executed this command
User	User ID for user who executed this command

Information	Description
Unique global ID	28-byte ID for user who owns this command
Buffer Addresses	buffer addresses for: control block, format buffer, search buffer, value buffer, ISN buffer
Buffer Lengths	FL: format buffer length RL: record buffer length SL: search buffer length VL: value buffer length IL: ISN buffer length
Snap Thread	The first 144 bytes of the user thread are snapped

Command Information

Information	Description
CQE Address	The address location of this CQE
F	Command queue flag bytes: ■ First Byte: General Purpose Flag ■ X'80': User buffers in service partition, region, address space ■ X'40': ET command waiting for 12 call ■ X'20': Waiting for 16 call ■ X'10': 16 call required ■ X'08': Attached buffer ■ X'04': Attached buffer required ■ X'02': X-memory lock held (z/OS only) ■ Second Byte: Selection Flag ■ X'80': In process ■ X'40': Ready to be selected ■ X'20': Search for UQE done ■ X'10': UQE found ■ X'08': Not selectable during BSS=x'80' status ■ X'04': Not selectable during ET-SYNC ■ X'02': Waiting for space ■ X'01': Waiting for ISN in HQ
CMD	The command type
File Number	The file number for this command
Job Name	Job name for the user
Addr User	UQE Address of users UQE, if searched for and found
Addr User ASCB	Address location of user's ASCB

Information	Description
Addr ECB	Address location of user's ECB (in user's address space)
Addr User UB	Address of users UB (in user's address space)
Addr User PAL	Address location of user's parameter address list
CQE ACA	ACA field of CQE.
CQE RQST	RQST field of CQE
Abuf/Pal	Address of the attached buffer/parameter address list (PAL) for CMD
Comm Id	28-byte unique user ID for this command

RABN Information

Information	Description
RABN Number	The RABN number in decimal
Type	Type of block (A - ASSO, D - DATA)
Flag	BP header element flag byte: ■ AKZ X'40': Active indicator ■ UKZ X'20': Update indicator ■ RKZ X'10': Read indicator ■ XKZ X'04': Access is waiting for block ■ YKZ X'02': Update is waiting for block ■ SKZ X'01': Write indicator
File	File number that owns this block
Address	Address location of block in storage.

10 Maintaining A Separate Test Environment

This section describes a method to set up a temporary test copy of phases updated by a program fix. The method described is intended as an example. Its relevance depends on the installation standards you use for library maintenance.

The example scenario uses MSHP in a single z/VSE machine to control both the standard production Adabas library and an additional testing library or sublibrary used to validate recently applied program fixes.

After restoring the standard Adabas library and defining it to MSHP, an additional test library or sublibrary can be defined.

Object modules can then be copied from the standard library as required, and controlled with MSHP using a different z/VSE system history file. Using the same component ID as for the standard environment (9001-ADA-00-*vrs*) ensures that the ZAP source remains common to both environments.

The test version of a phase is then invoked by placing the test library or sublibrary at the head of the LIBDEF PHASE search chain.

The setup jobs required to implement this environment are described in detail below. Note that the first three steps form part of the standard installation process.

to setup the separate test environment:

1 Define standard Adabas library.

For a sample job, see the section ***Installing the Adabas Release Tape***.

2 Restore standard Adabas library.

For a sample job, see the section ***Installing the Adabas Release Tape***.

3 Define standard Adabas to MSHP.

Note: This job uses the history file identified by the IJSYSHF label in the z/VSE standard label area.

```
// EXEC MSHP
ARCHIVE ADAvrs
COMPRISES 9001-ADA-00
RESOLVES 'SOFTWARE AG - ADABAS.ADAvrs'
ARCHIVE 9001-ADA-00-vrs
RESIDENCE PRODUCT=ADAvrs -
PRODUCTION=SAGLIB.adannn -
GENERATION=SAGLIB.adannn
/*
```

where *vrs* is the Adabas version, revision, and system maintenance (SM) level and *adannn* is the sublibrary name for standard Adabas.

4 Create test sublibrary and copy object modules to it.

```
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// EXEC LIBR
DEFINE SUBLIB=SAGLIB.adatst
CONNECT SAGLIB.adannn:SAGLIB.adatst
COPY *.OBJ LIST=Y REPLACE=Y
/*
```

where *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, *adannn* is the sublibrary name for standard Adabas, and *adatst* is the sublibrary name for testing Adabas.

5 Create additional system history file for test environment and define test Adabas to it.

```
// ASSGN SYS020,DISK,VOL=volhis,SHR
// EXEC MSHP
CREATE HISTORY SYSTEM
DEFINE HISTORY SYSTEM EXTENT=start:numtrks -
UNIT=SYS020 -
ID='sag.test.system.history.file'
ARCHIVE ADAvrs
COMPRISES 9001-ADA-00
RESOLVES 'SOFTWARE AG - ADABAS Vvrs'
ARCHIVE 9001-ADA-00-vrs
RESIDENCE PRODUCT=ADAvrs -
PRODUCTION=SAGLIB.adatst -
GENERATION=SAGLIB.adatst
/*
```

where *volhis* is the volume on which test system history file resides, *start* is the start of extent on which test system history file resides, *numtrks* is the length of extent on which test system history file resides, *sag.test.system.history.file* is the physical name of test system history file, *vrs* is the Adabas *version*, and *adatst* is the sublibrary name for testing Adabas.

6 Apply zap to test environment.

```
// DLBL IJSYSHF,'sag.test.system.history.file'
// EXTENT SYS020,,,,start,numtrks
// ASSGN SYS020,DISK,VOL=volhis,SHR
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// EXEC MSHP
CORRECT 9001-ADA-00-vrs : ADnnnnn
AFFECTS MODULE=modname
ALTER offset hexold : hexnew
INVOLVES LINK=lnkname
/*
```

where *sag.test.system.history.file* is the physical name of test system history file, *start* is the start of extent on which test system history file resides, *numtrks* is the length of extent on which test system history file resides, *volhis* is the volume on which test system history file resides, *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, *vrs* is the Adabas *version*, *nnnnn* is the Adabas fix number, *modname* is the Adabas object module to be zapped and then relinked, *offset* is the hexadecimal offset to the beginning of the zap, *hexold* is the verify data for the zap, *hexnew* is the replace data for the zap, and *lnkname* is the link book for the phase affected.

7 Invoke updated test phase.

```
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// LIBDEF PHASE,SEARCH=(SAGLIB.adatst,SAGLIB.adannn,...)
...
```

where *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, *adatst* is the sublibrary name for testing Adabas, and *adannn* is the sublibrary name for standard Adabas.

8 Apply zap to standard environment.

Note: This job uses the history file identified by the IJSYSHF label in the z/VSE standard label area.

```
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// EXEC MSHP
CORRECT 9001-ADA-00-vrs : ADnnnnn
AFFECTS MODULE=modname
ALTER offset hexold : hexnew
INVOLVES LINK=lnkname
/*
```

where *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, *vrs* is the Adabas *version*, *nnnnn* is the Adabas fix number, *modname* is the Adabas object module to be zapped and then relinked, *offset* is the hexadecimal offset to the beginning of the zap, *hexold* is the verify data for the zap, *hexnew* is the replace data for the zap, and *lnkname* is the link book for the phase affected.

9 Invoke standard phase.

```
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// LIBDEF PHASE,SEARCH=(SAGLIB.adannn,...)
...
```

where *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, and *adannn* is the sublibrary name for standard Adabas.

11 Translation Tables

This section describes the translation tables which are supplied by Adabas.

Adabas EBCDIC to ASCII and ASCII to EBCDIC

```
cUES2ASC DS 0F
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c DC x'000102033F093F7F3F3F3F0B0C0D0E0F' 0.
c DC x'101112133F3F083F18193F3F3F1D3F1F' 1.
c DC x'3F3F1C3F3F0A171B3F3F3F3F3F050607' 2.
c DC x'3F3F163F3F1E3F043F3F3F3F14153F1A' 3.
c DC x'203F3F3F3F3F3F3F3F3F3F2E3C282B3F' 4.
c DC x'263F3F3F3F3F3F3F3F3F21242A293B5E' 5.
c DC x'2D2F3F3F3F3F3F3F3F3F7C2C255F3E3F' 6.
c DC x'3F3F3F3F3F3F3F3F3F603A2340273D22' 7.
c DC x'3F6162636465666768693F3F3F3F3F3F' 8.
c DC x'3F6A6B6C6D6E6F7071723F3F3F3F3F3F' 9.
c DC x'3F7E737475767778797A3F3F3F5B3F3F' A.
c DC x'3F3F3F3F3F3F3F3F3F3F3F3F3F5D3F3F' B.
c DC x'7B41424344454647484 93F3F3F3F3F3F' C.
c DC x'7D4A4B4C4D4E4F5051523F3F3F3F3F3F' D.
c DC x'5C3F535455565758595A3F3F3F3F3F3F' E.
c DC x'303132333435363738393F3F3F3F3F3F' F.
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END

cUES2EBC DS 0F
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c DC x'000102037372D2E2F160525 0B0C0D0E0F' 0.
c DC x'101112133C3D322618193F27221D351F' 1.
c DC x'405A7F7B5B6C507D4D5D5C4E6B604B61' 2.
c DC x'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F' 3.
c DC x'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' 4.
c DC x'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D' 5.
c DC x'7981828384858687888991929 3949596' 6.
c DC x'979899A2A3A4A5A6A7A8A9C06AD0A107' 7.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' 8.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' 9.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' A.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' B.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' C.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' D.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' E.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' F.
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END
```

Entire Net-Work EBCDIC to ASCII and ASCII to EBCDIC

```
NW2ASC DS OF
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
DC X'000102030405060708090A0B0C0D0E0F' 0.
DC X'101112131415161718191A1B1C1D1E1F' 1.
DC X'00000000000000000000000000000000' 2.
DC X'00000000000000000000000000000000' 3.
DC X'2000000000000000000005B2E3C282B5D' 4.
DC X'26000000000000000000021242A293B5E' 5.
DC X'2D2F0000000000000007C2C255F3E3F' 6.
DC X'00000000000000000603A2340273D22' 7.
DC X'00616263646566676869000000000000' 8.
DC X'006A6B6C6D6E6F707172000000000000' 9.
DC X'007E737475767778797A00005B000000' A.
DC X'0000000000000000000000000005D0000' B.
DC X'7B41424344454647484900000000000' C.
DC X'7D4A4B4C4D4E4F505152000000000000' D.
DC X'5C7E535455565758595A000000000000' E.
DC X'303132333435363738397C00000000FF' F.
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F

NW2EBC DS OF
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
DC X'000102030405060708090A0B0C0D0E0F' 0.
DC X'101112131415161718191A1B1C1D1E1F' 1.
DC X'405A7F7B5B6C507D4D5D5C4E6B604B61' 2.
DC X'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F' 3.
DC X'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' 4.
DC X'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D' 5.
DC X'79818283848586878889919293949596' 6.
DC X'979899A2A3A4A5A6A7A8A9C06AD0A100' 7.
DC X'00000000000000000000000000000000' 8.
DC X'00000000000000000000000000000000' 9.
DC X'00000000000000000000000000000000' A.
DC X'00000000000000000000000000000000' B.
DC X'00000000000000000000000000000000' C.
DC X'00000000000000000000000000000000' D.
DC X'00000000000000000000000000000000' E.
DC X'000000000000000000000000000000FF' F.
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END
```

Index

M

N

O

P

R

S

T

U

X

Z