Installing Adabas for z/OS

This chapter describes the installation of Adabas on z/OS systems.

- Installation Checklist
- Contents of the Release Tape
- Preparing to Install Adabas
- Initializing the Adabas Communication Environment
- Installing an Adabas Database
- SVC Integrity Validation
- Requirements for Cross-Memory Services
- Requirements for Global Resource Serialization
- Using EXCPVR
- Creating a Shareable ADARUN
- Storage Above 16 MB
- Storage Above 2 GB (64-Bit)
- Applying Zaps
- Adabas 8 Adalink Considerations

Installation Checklist

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

Important:
Be sure that you apply all supplied Adabas 8 maintenance and concatenate Adabas 8 patch-level libraries (L00*n*), as they are delivered to you. This will ensure that your Adabas 8 code remains up-to-date, supporting all Adabas 8 features as they are enhanced and maintained.

Step	Description	Additional Information
	Basic Installation Steps	
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*.

Step	Description	Additional Information
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 the address space for running the Adabas nucleus.	Refer to the section *Adabas Nucleus Address Space Requirements*.
4	Restore the Adabas libraries from the installation tape.	Use the tape positioning information that accompanies the tape. Refer to the section *Installing the Release Tape*.
5	Install the Adabas SVC temporarily or permanently.	Refer to the section *Initializing the Adabas Communication Environment*.
	Database Installation Steps	Steps 6-15 require changes to the setup definitions as described in section *Installing an Adabas Database*.
6	Allocate and format the Adabas database with the ADAFRM utility job.	
7	Define the global database characteristics with the ADADEF utility job.	
8	Load the demonstration files with the ADALODE, ADALODV, ADALODM, and ADALODP jobs.	
9	Prepare and install the product license file.	
10	Customize and start the Adabas nucleus and test the Adabas communications with the appropriate ADANUC job.	
11	If appropriate, test Adabas address space communications by running ADAREP in MULTI mode with the CPEXLIST parameter.	
12	If appropriate, load the Adabas Online System (AOS) add-on product into a Natural system file by running the ADAINPL job. Alternatively, install the AOS demo version (see section *Installing the AOS Demo Version*).	
13	Terminate the Adabas nucleus with an ADAEND operator command using the OS Modify command F.	
14	Back up the database by running the ADASAV utility job.	
15	Insert the ADARUN defaults by running the DEFAULTS job.	
	TP Monitor Installation	

Step	Description	Additional Information
16	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/OS environments.

Library Name	Description
ACI*vrs*.LOAD	The load library for the Adabas CICS link routine. The *vrs* in the library name represents the *version* of Adabas.
ACI*vrs*.SRCE	The source library for the Adabas CICS link routine. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.EC*nn*	The Adabas library containing character encoding members to support various languages and Unicode. The *vrs* in the library name represents the *version* of Adabas. The *nn* letters in the library name represents a number from "00" to "99", assigned by Software AG.
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*.JOBS	The sample job library for Adabas. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.LOAD	The load library for Adabas. The *vrs* in the library name represents 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 new Personnel demo file. The *vrs* in the library name represents the *version* of Adabas.

Library Name	Description
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 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*.SRCE	The source library for Adabas. The *vrs* in the library name represents the *version* of Adabas. For a complete list of the time zones supported by Adabas in any given release, refer to the TZINFO member in this Adabas library.
ADA*vrs*.TZ00	The time zone library for Adabas. The *vrs* in the library name represents the *version* of Adabas. Adabas bases its time zone library on the time zones defined in the public domain tz database, also known as the *zoneinfo* or *Olson* database. For a complete list of the time zones supported by Adabas in any given release, refer to the TZINFO member in the Adabas source library (ADA*vrs*.SRCE).
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.
AII*vrs*.LOAD	The load library for the Adabas IMS link routine. The *vrs* in the library name represents the *version* of Adabas.
AII*vrs*.SRCE	The source library for the Adabas IMS link routine. The *vrs* in the library name represents the *version* of Adabas.
APS*vrs*.LD*nn*	One or more Software AG internal libraries. 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*.JOBS	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*.LOAD	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*.JOBS	The sample job 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.

Library Name	Description
WAL*vrs*.LOAD	The load 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.
WAL*vrs*.SRCE	The source 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.

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.

This section covers the following topics:

- Disk Space Requirements for Libraries
- Data Sets Required for UES Support
- Disk Space Requirements for Internal Product Data Sets
- Disk Space Requirements for the Database
- Adabas Nucleus Address Space Requirements

Disk Space Requirements for Libraries

The minimum 3390 disk space requirements for the Adabas libraries are as follows:

Library	3390 Cylinders	3390 Tracks	Directory Blocks
Load	19	285	30
Source	7	105	20
JCL	1	15	20

Note:
You can isolate user programs from the Adabas load library by creating a separate load library that contains only those modules needed to execute user programs in multi-user mode and linked with ADAUSER. For this Adabas version, the modules required by user programs are ADAIOR, ADAIOS, ADALNK, ADAMLF, ADAPRF, ADARUN.

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 delivered separately from the product libraries.

For UES support, the following library must be loaded and included in the STEPLIB concatenation, where *nn* is the load library level and *vrs* is the number of the latest *version* of that code delivered on the tape:

APS*vrs*.LD*nn*

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 STEPLIB concatenation.

Note:
If you are using an Adabas load library prior to Version 7.2.2, it contains internal product libraries with an earlier version number and must be ordered below the current internal product libraries in the STEPLIB concatenation.

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

ADA*vrs*.EC*nn*

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

Disk Space Requirements for Internal Product Data Sets

The minimum disk space requirements on a 3390 disk for the internal product libraries delivered with this version of Adabas are as follows (where *vrs* is the latest *version* of the code delivered on the tape):

Library	3390 Cylinders	3390 Tracks	Directory Blocks
ADA*vrs*.EC*nn*	23	345	200
APS*vrs*.LD*nn*	6	86	60

Disk Space Requirements for the Database

The actual database space needed by Adabas depends on user requirements. The minimum 3390 disk space requirements for the database are as follows:

Database Component	3390 Cylinders	3390 Tracks
ASSOR1 (Associator)	20	300
DATAR1 (Data Storage)	60	900
WORKR1 (Work space)	15	225
TEMPR1 (temporary work space)	15	225
SORTR1 (sort work space)	15	225

Adabas Nucleus Address Space Requirements

The typical Adabas nucleus requires at least 800-1024 kilobytes to operate. The size of the nucleus address space may need to be larger, depending on the ADARUN parameter settings. Parameter settings are determined by the user.

Initializing the Adabas Communication Environment

This section describes the installation of the Adabas router (ADASVC). The router uses cross-memory services for communication between the Adabas nucleus and the Adabas users.

The Adabas z/OS cross-memory communications service comprises two modules:

- the Adabas router (ADASVC); and
- the Adabas subsystem initialization routine (ADASIR).

ADASIR, executed either during IPL or by the Adabas SVC installation program (ADASIP), initializes the router's operating environment, particularly the ID table.

ADASVC installation can be either temporary or permanent:

- The Adabas SVC can be installed temporarily by executing ADASIP. The SVC is then available only until the next IPL.

  Note:
  Once installed, the Adabas SVC can be re-installed temporarily using the ADASIP REPLACE option. However, no Adabas nucleus can be active during this procedure.

  Note:
  It is necessary to cycle CICS after executing ADASIP to initialize the SVC.

- The Adabas SVC is installed permanently using regular operating systems procedures. The SVC then requires an IPL to become active.

Typically, the Adabas SVC is first installed temporarily using ADASIP. This makes Adabas available immediately without the need to wait for an IPL. Meanwhile, preparations are usually made for permanent installation at the next IPL.

- SVC Compatibility Issues Between Adabas Releases
- Authorization Requirements
- Allocating an SVC Table Entry
- Subsystem Name Requirements
- Page-Fixing the Adabas SVC
- Initializing the Adabas SVC

- Router Installation Overview
- Using ADASIP for Temporary Installations
- Using ADASIR
- Relinking the SVC for Temporary Installation
- Relinking the SVC for Permanent Installation

SVC Compatibility Issues Between Adabas Releases

Adabas 8 includes a new Adabas SVC. This SVC is fully backward compatible. In other words, you can use the new Adabas 8 SVC with Adabas 7 (or earlier) databases.

However, you *cannot* use the Adabas SVC from previous Adabas releases with Adabas 8 databases. If you attempt to do this, the Adabas 8 database will not initialize successfully.

The Adabas 8.2 SVC includes performance improvements and improved error recovery routines. Note that the new SVC uses more efficient operating system interfaces, in particular when posting the user at command completion. This shifts work from SRB-mode routines to TCB-mode routines and also between the user's program and the Adabas nucleus. Take this into account when analyzing Adabas 8 SVC performance. With the new SVC, SRB-mode overhead is largely eliminated and TCB-mode overhead is somewhat increased, but the net result is an overall improvement in SVC performance.

Authorization Requirements

The Adabas 8.2 (and later) SVC requires that the Adabas nucleus, as well as other MPM servers (such as Entire Net-Work and the Natural Global Buffer Pool), be authorized to prevent inappropriate use of critical ADASVC functions. This APF authorization prevents unauthorized use of the ADASVC 0-call. Software AG recommends strongly that you run APF-authorized because of the security risks you can incur if you do not. However, upon request, Software AG does have a zap you can apply that eliminates this requirement. To determine what this zap number is, review the ZAPOPT member in the Adabas source library for a zap entitled "Remove requirement for APF authorization".

Note:
Some add-on products require APF authorization to use restricted z/OS services. APF authorization is still required in these cases.

There are two authorization mechanisms: System Authorization Facility (SAF) and APF authorization. SAF is the z/OS standard interface to security products such as RACF, ACF2, and Top Secret. APF is the z/OS facility that allows programs from designated libraries to access restricted z/OS functions.

The SAF authorization check occurs first. It requests read access to an entity in the FACILITY class. For a classic Adabas nucleus, the entity is of this form:

`ADABAS.SVC`*`sss`*`.ID`*`ddddd`*

The *sss* is the Adabas SVC number and *ddddd* is the DBID with leading zeros.

If the nucleus is using Adabas Cluster Services or Adabas Parallel Services, the entity has an additional level:

`ADABAS.SVC`*`sss`*`.ID`*`ddddd`*`.Nuc`*`nnnnn`*

The *nnnnn* is the value assigned by the ADARUN NUCID parameter with leading zeros.

The SAF security administrator can assign permissions to entities of this form with a wide range of specificity using a flexible array of patterning and wildcard characters. In general, it should not be necessary to enumerate each possible combination of SVC, DBID, and NUCID.

There are three possible outcomes to the SAF check: permission is explicitly granted, permission is explicitly denied, or the entity may be unknown to SAF. When the entity is unknown, APF authorization is required.

APF Authorization Status	SAF Allow	SAF Deny	Unknown to SAF
APF authorized	Allow	ABEND U494	Allow
Not APF authorized	Allow	ABEND U494	ABEND U494

Message ADAS33 is issued before any ABEND U494.

Note:

APF authorization is still required if your Adabas nucleus configuration needs to use restricted z/OS services. For example, the EXCPVR option and some add-on products such as Adabas Cluster Services and Adabas SAF Security (ADASAF) cannot run without APF authorization. IN these cases, ADASVC SAF may be used only to deny permission to initialize.

The ADASAF add-on product can also restrict access to a DBID/SVC combination (but not an Adabas Cluster Services or Adabas Parallel Services nucleus ID). ADASAF uses a different resource class and arranges the subfields in resource entities differently. You may choose to use either SAF mechanism, both, or neither. To maintain their existing behavior, current ADASAF users should not create ADABAS.SVC* resource profiles in the FACILITY class and need not change anything in their existing ADASAV configuration.

APF authorization requires that all libraries in the JOBLIB and STEPLIB concatenations have entries in the z/OS APF list. Systems programming staff typically administer the APF list.

Allocating an SVC Table Entry

Regardless of the installation procedure selected, an available SVC table entry must be allocated to the Adabas router (ADASVC). SVC table entries are defined in the member IEASVCxx of SYS1.PARMLIB.

The SVC table entry in the operating system for an ADASVC must contain the following information:

Offset	Label	Description
0	SVCEP	SVC entry point address.
4	SVCATTR1	Must indicate type 2 SVC (flag bit SVCTP2 set—X'80') or type 3 or 4 SVC (flag bits SVCTP34 set—X'C0'): ADASIR changes a type 1, 5, or 6 SVC to type 2. May indicate that APF-authorization is needed for this SVC (flag bit SVCAPF set—X'08'): if set, all targets and users must be APF-authorized.
6	SVCLOCKS	Must contain all zeros. ADASIR sets SVCLOCKS to zeros.

Subsystem Name Requirements

The subsystem name contained in the four-character field SUBSYS at ADASVC offset X'28' (the default is "ADAB") must be the same as that specified in the IEFSSN*xx* member of SYS1.PARMLIB. If the name is not the same, ADASIR ends with an ADAS12 message and condition code 2, and Adabas is not usable.

Page-Fixing the Adabas SVC

If the Adabas SVC is to reside in the fixed LPA, add an entry to an IEAFIX*xx* member of SYS1.PARMLIB.

Initializing the Adabas SVC

The Adabas SVC should be initialized with ADASIP/ADASIR in order to guarantee full functioning of all Adabas nuclei.

Router Installation Overview

Temporary Router Installation (SMA Job Number I011)

Once you have restored the Adabas installation tape, use a local editor to customize the job JCLLINK (used to link ADASIR, ADASIP, and ADASVC) as follows:

to perform temporary router installation:

1. Link ADASIP into an APF-authorized library as an authorized module.
2. Link ADASIR and ADASVC into APF-authorized libraries:
   - Place ADASVC in an APF-authorized library in order to run ADASIP.
   - Place ADASIR in an APF-authorized library.
3. Execute ADASIP to install the SVC.

   Customize and run the job ADASIP to dynamically add the Adabas SVC without an IPL.

Permanent Router Installation (SMA Job Number I010)

to perform permanent router installation:

1. Link the Adabas SVC (ADASVC) which has been renamed according to the SVC routine renaming rules (for example, type 3 SVCs must have names of IGC00*nnn*, where *nnn* is a signed decimal SVC number) into SYS1.LPALIB as a permanent step for ADASIR.
2. Link ADASIR into SYS1.LINKLIB or into an APF-authorized library concatenated to SYS1.LINKLIB with the LNKLST*xx* member of SYS1.PARMLIB.

   Note:
   ADASIR is not reentrant, and therefore should not be linked into SYS1.LPALIB.
3. Customize and run the job JCLUPDT to add a new entry with the correct format.
4. IPL z/OS with the CLPA option to install and initialize the Adabas communication environment.

Using ADASIP for Temporary Installations

ADASIP Functions

ADASIP performs the following functions:

- acquires memory in the specified CSA subpool for the Adabas SVC and a subsystem communication vector table (SSCT)
- loads the Adabas SVC into the acquired CSA space
- modifies the SVC table entry as required by the Adabas SVC
- optionally deletes an SSCT for the same subsystem name from the SSCT chain
- adds the new SSCT to the SSCT chain
- invokes the ADASIR program
- releases CSA acquired by a previously installed SVC

If any error is detected, ADASIP backs out all completed activities and terminates operation with a user abend specifying the error.

When reinstalling an instance of ADASVC using an SVC number that is currently being used by ADASVC, the subsystem name must be the same as the one currently being used. This helps avoid a configuration that may not function correctly. For more information, read *SVC Integrity Validation*.

A Version 8 ADASIP/ADASIR can be used to install a Version 7 Adabas SVC, but the Version 8 SVC requires the Version 8 ADASIP/ADASIR. Software AG recommends using Version 8 to install all supported releases of the SVC. If an earlier version of ADASIP/ADASIR is used to replace an SVC installed with a later version, some areas of common storage may not be released.

The following JCL links ADASIP, located in ADABAS.ADA*vrs*.LOAD, into an APF-authorized library as an authorized module:

```
//LNKSIP EXEC PGM=IEWL
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD SPACE=(CYL,(1,1)),UNIT=SYSDA
//ADALIB DD DSN=ADABAS.ADAvrs.LOAD,DISP=SHR
//SYSLMOD DD DSN=apflibname,DISP=SHR
//SYSLIN DD *
INCLUDE ADALIB(ADASIP)
SETCODE AC(1)
NAME ADASIP(R)
```

ADASIP Parameters

ADASIP parameters have the following syntax:

```
CONSNAME=c,IDTSPL=i,LEAVE=l,NRIDTES=n,REPLACE=r,SUBSYS=su,
SVCNR=svcn,SVCSPL=svcs
```

—where

c	is the console name to which operator messages are written. If omitted, messages are issued using ROUTCDE=2, master Console Information.
i	is the ID table subpool: see the ADASIR IDTSPL parameter for details.
l	indicates whether ADASIR should display message ADAS11 or ADAS12 on the operator console: see the ADASIR LEAVE parameter for details.
n	is the number of ID table entries: see the ADASIR NRIDTES parameter for details.
r	indicates whether or not an existing SSCT for the same subsystem name is to be replaced. Y for yes or N for no (N is the default). Use this option to replace any type of Adabas SVC (for example, when installing a new SVC version).
su	is the subsystem name. This parameter is required. . Each instance of the Adabas SVC must have a unique subsystem name.
svcn	is the Adabas SVC number: see the ADASIR SVCNR parameter for details.
svcs	is the Adabas SVC and SSCT subpool: 228 for fixed CSA or 241 for pageable CSA (default: 241).

The following are valid ADASIP parameter abbreviations:

Parameter	Abbreviation
CONSNAME=	C=
IDTSPL=	I=
LEAVE=	L=
NRIDTES=	N=
REPLACE=	R=
SUBSYS=	SU=
SVCNR=	SVCN=
SVCSPL=	SVCS=

All parameters are optional except SUBSYS and SVCNR. If specified, the parameters IDTSPL,LEAVE, NRIDTES, SUBSYS, and SVCNR are passed to ADASIR without being verified.

Executing ADASIP

JCL similar to the following should be used to execute ADASIP:

```
// EXEC PGM=ADASIP,PARM=parameters
//STEPLIB DD ...
//SVCLIB DD ...
//SIRLIB DD ...
```

The data set defined by the STEPLIB DD statement must be an APF-authorized library containing the APF-authorized program ADASIP. Since ADASIP is neither reentrant nor refreshable, the data set cannot be SYS1.LPALIB.

The data set defined by the SVCLIB DD statement must be an APF-authorized library containing the Adabas SVC with either the name or alias ADASVC.

The data set defined by the SIRLIB DD statement must contain the ADASIR program. Since ADASIR is neither reentrant nor refreshable, the data set may not be SYS1.LPALIB.

ADASIP terminates with a U0481 abend if the parameter input is incorrectly specified.

The IBM job control convention for continuing the PARM parameter is:

```
// EXEC PGM=ADASIP,PARM=('parameters ....', X
// 'parameters')
```

—where X in column 72 is a continuation character. The following restrictions also apply to JCL statements:

- a comma is required after the end-quote on a line that is to be continued
- a non-blank continuation character is required in column 72 of each line that is to be continued, and the continuation line must start within columns 4-16

- a comma is not permitted between the last parameter and the end-quote on the line to be continued because JCL automatically inserts a comma between parameters when concatenating continuation strings:

  ○

  ```
  // ...PARM=('CONSID=3', X
  // 'SUBSYS=ADAB', X
  // 'SVCNR=249')
  ```

  —results in an equivalent line of

  ```
  CONSID=3,SUBSYS=ADAB,SVCNR=249
  ```

Using ADASIR

ADASIR Functions

The ADASIR program is invoked

- by the ADASIP program to install the Adabas SVC temporarily, or
- by z/OS to install the Adabas SVC permanently.

ADASIR receives control during either master scheduler initialization or ADASIP execution. The operator is prompted for any value that has been incorrectly zapped or assembled (refer to the *Adabas Messages and Codes* for specific message descriptions). If an error is found during the processing of parameters specified in the IEFSSN*xx* member or passed by ADASIP, the operator is prompted for all of the values.

If the SVC table entry is incorrect, ADASIR prompts the operator for permission to change the entry (if SVCTAB=P, the default, is specified). If any errors are detected, they must be corrected and either another IPL must be done or ADASIP must be rerun before the Adabas SVC can be used.

Relinking ADASIR

The ADASIR module must be linked into an APF-authorized library.

The following JCL links ADASIR, located in ADABAS.ADA*vrs*.LOAD, into SYS1.LINKLIB:

```
//LNKSIR EXEC PGM=IEWL
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD SPACE=(CYL,(1,1)),UNIT=3390
//ADALIB DD DSN=ADABAS.ADAvrs.LOAD,DISP=SHR
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSLIN DD *
INCLUDE ADALIB(ADASIR)
SETCODE AC(1)
NAME ADASIR(R)
```

ADASIR Parameters

ADASIR parameters have the following syntax:

IDTSPL=*i*,LEAVE=*l*,NRIDTES=*n*,SVCNR=*svcn*,SVCTAB=*svct*

Variable	Description
i	The ID table subpool: 228 for fixed CSA or 241 (the default) for pageable CSA.
l	Indicates whether message ADAS11 or ADAS12 is to be displayed on the operator console: Y for yes or N (the default) for no.
n	The ID table entry count, which can range from 1 to a maximum specified at offset X'146' in the CSECT IEAVESVT of the z/OS nucleus (see section *Requirements for Cross-Memory Services*).
svcn	The Adabas SVC number (200-255).
svct	Indicates whether or not the operator should be prompted for permission to update the SVC table entry. Enter P (the default) to receive a prompt, or N for no prompt. P is recommended if a possibility exists that the SVC table entry will not be what ADASIR expects.

The following are valid abbreviations for ADASIR parameters:

Parameter	Abbreviation
IDTSPL=	I=
LEAVE=	L=
NRIDTES=	N=
SVCNR=	SVCN=
SVCSPL=	SVCS=

Executing ADASIR

Note:
The ADASIR module must be linked into an APF-authorized library.

To prepare for permanent SVC installation, an entry must be made in either a new or existing member having the name IEFSSN*xx* in SYS1.PARMLIB. This entry is an 80-character record with the following format:

```
SUBSYS SUBNAME(cccc) CONSNAME(consname) INITRTN(ADASIR) INITPARM('parameters') comments
```

—where

cccc	The 1- to 4-character subsystem name. This name and the name specified in the Adabas SVC at offset X'28' must be the same. The name provided in the SVC is ADAB; any other name must first be zapped into the SVC before being specified for *cccc*.
consname	The name of the console to which ADASIR will direct any messages. If omitted messages will be issued with ROUTCDE=2, Master Console Information.
'parameters'	ADASIR parameters. If there is more than one parameter, values must be enclosed in single quotation marks and a comma placed between the parameters.
comments	Comments are optional and must be preceded by at least one space.

If the subsystem name does not match, ADASIR abends with an ADAS12 message and condition code 2; the Adabas z/OS communication environment is not initialized. Re-IPL z/OS, specifying SSN=*xx* if necessary. If this is the first IPL with a type 3 or 4 Adabas SVC, specify CLPA as one of the SET parameters.

If an error is encountered while processing any of the parameters obtained from the IEFSSN*xx* member or passed from ADASIP (message ADAS05), the operator is prompted to reenter all of the parameters. If the SVC table entry is not correct (message ADAS09) then, depending on the value of the SVCTAB parameter, either the operator is prompted (message ADAS10) for permission to change the SVCTAB parameter, or it is simply changed (message ADAS15).

A Version 8 ADASIR can be used to install a Version 7 Adabas SVC, but the Version 8 SVC requires the Version 8 ADASIR. Software AG recommends using Version 8 to install all supported releases of the SVC.

The ADASIR messages and their meanings are described in the *Adabas Messages and Codes*.

Relinking the SVC for Temporary Installation

Link the Adabas SVC with the name or alias ADASVC into an APF-authorized library. ADASVC must be linked with AMODE=31 and RMODE=24, the default.

The following example shows how to link the SVC:

```
// (job card)
//LKED EXEC PGM=IEWL,
// PARM='XREF,LIST,NCAL,LET,MAP,RENT,REUS'
//SYSPRINT DD SYSOUT=X
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR <--Target loadlib
//ADALIB DD DSN=user.loadlib,DISP=SHR <--ADASVC loadlib
//SYSLIN DD *
MODE AMODE(31),RMODE(24)
INCLUDE ADALIB(ADASVC)
NAME ADASVC(R)
/*
```

Note:
If the SVC is linked with a name other than ADASVC when preparing to upgrade to a permanent installation, the SVC must have an alias of ADASVC. When dynamically loading the Adabas SVC, ADASIP searches for the module ADASVC in the library specified by the SVCLIB DD statement.

Relinking the SVC for Permanent Installation

Software AG recommends using a type 3 or 4 SVC for the Adabas SVC.

SVC types 1 and 6 are not supported.

- Type 2 SVC
- Type 3 or 4 SVC

Type 2 SVC

If the Adabas SVC is to be type 2, link it into SYS1.NUCLEUS as the system nucleus IEANUC0*x*.

This nucleus must contain an SVC table entry for an enabled type 2 SVC, which must be defined during SYSGEN.

Then include linkage editor control statements similar to the following with those needed to link a nucleus:

```
CHANGE ADASVC(IGCnnn) ---> nnn is the SVC number in decimal
INCLUDE ADALIB(ADASVC) ---> ADALIB contains the Adabas SVC
```

Type 3 or 4 SVC

To install the Adabas SVC as type 3 or 4, link the Adabas SVC with the appropriate name into SYS1.LPALIB. ADASVC must be linked with AMODE=31 and RMODE=24 (the default).

The following example shows how to relink the SVC:

```
// (job card)
//LKED EXEC PGM=IEWL,
// PARM='XREF,LIST,NCAL,LET,MAP,RENT,REUS'
//SYSPRINT DD SYSOUT=X
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD DD DSN=SYS1.LPALIB,DISP=SHR <--Target Loadlib
//ADALIB DD DSN=ADABAS.ADAvrs.LOAD,DISP=SHR <--ADASVC loadlib
//SYSLIN DD *
MODE AMODE(31),RMODE(24)
CHANGE ADASVC(IGC00nnp) <- where nn are the first two digits of the SVC in decimal and
INCLUDE ADALIB(ADASVC) p is the character corresponding to the x'Cn'
NAME IGC00nnp(R) ----> (n is the last digit of the SVC number in decimal)
*
/*
```

Installing an Adabas Database

Once you have installed the Adabas installation tape and have initialized the ADASVC, you can:

- Migrate an existing Adabas database to the new version; or
- Install a new version of the Adabas database.

Messages or codes that occur during the installation are described in the *Adabas Messages and Codes*; utilities are described in the *Adabas Utilities*.

- Migrate an Existing Database
- Installing the Adabas Release Tape
- Database Installation Steps

Migrate an Existing Database

Use the ADACNV utility to migrate existing databases to new releases of Adabas (SMA job number I021). See *Adabas Utilities* for more information.

Installing the Adabas Release Tape

Note:
If you are using System Maintenance Aid (SMA), refer to the *System Maintenance Aid* documentation. If you are not using SMA, follow the instructions below.

This section explains how to copy all data sets from tape to disk. You will then need to perform the individual installation procedure for each component to be installed.

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

Step 1: Copy Data Set COPY.JOB from Tape to Disk

The data set `COPY.JOB` contains the JCL required to copy all data sets from tape to disk. If the data sets for more than one product are delivered on the tape, the data set `COPY.JOB` contains the JCL to unload the data sets for all delivered products from the tape to your disk.

Copy `COPY.JOB` to your disk using the following sample JCL:

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

where:

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

Step 2: Modify COPY.JOB on Your Disk

Modify `COPY.JOB` according to your local naming conventions and set the following disk space parameters:

- Set `HILEV` to a valid high-level qualifier.
- Set `LOCATION` to a storage location.
- Set `EXPDT` to a valid expiration date.

Step 3: Submit COPY.JOB

Submit `COPY.JOB` to copy all data sets from tape to your disk.

Database Installation Steps

Step 1: Check, Prepare, and Install the Product License File (SMA job number I007)

You must install a valid license file on all mainframe platforms in which your Software AG mainframe product is installed. The license file is provided as an XML document (encoding is US-ASCII) and must remain in that format -- even on the mainframe. It must not be modified. Any modification of the license file will invalidate the digital signature and the license check will fail. In the event of a check failure, please contact your Software AG technical support representative.

Note:
Thirty days before the license expires, license check failure messages are produced. Your software product will still function, but these messages warn you that it is time to obtain a new license.

In this step, you will prepare the license file (obtain it from e-mail or the installation tape and store it on your z/OS system) and then install it:

- Preparing the Product License File
- Installing the Product License File

Preparing the Product License File

The product license file is supplied on the individual customer installation tape or separately via an e-mail attachment. Before you can install the license, you must transfer it from e-mail or the installation tape and store it on a z/OS system. This section describes how to do this for a license distributed either by e-mail or on the installation tape.

To prepare the license file from an e-mail attachment, complete the following steps:

1. Transfer the license to z/OS, as described in *Transferring a License File from PC to a z/OS Host Using FTP*.
2. Verify that the transferred 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.

To prepare the license file from the installation tape, complete the following steps:

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

Installing the Product License File

Once the license file has been prepared, you can install it in one of two ways:

- You can convert the license to a load module (ADALIC) that is then loaded by the Adabas nucleus.
- You can reference the license file in the Adabas nucleus startup job by DD statement.

This section describes both methods.

To convert the license file to a load module, complete the following steps:

1. Review and modify sample job ASMLICAM, as follows:
   - Change the STEPLIB DD statement to point to the license load library (MLC*vrs*.LOAD).
   - Change the SYSUT1 DD statement to point to the data set containing the Adabas license file you transferred to z/OS earlier.
   - Specify an appropriate user load library for the L.SYSLMOD DD statement.

     Note:
     This user load library must also be included in the STEPLIB concatenation for the Adabas nucleus (ADANUC).
2. Submit sample job ASMLICAM. This job runs the MAKE function of the LICUTIL utility to convert the license text file to an assembler source module. ASMLICAM then links and assembles the assembler source to generate a load module called ADALIC, which is stored in the specified user load library (L.SYSLMOD DD statement). For more information about the LICUTIL utility, read *Using The License Utility: LICUTIL*
3. Update your Adabas ADANUC nucleus job to reference the user load library so ADALIC will be loaded by the Adabas nucleus, as described in *Step 5: Customize and Start the Adabas Nucleus and Test Adabas Communications*.

To reference the license file in the Adabas nucleus startup job, complete the following steps:

1. Make sure any previously created ADALIC load module is inaccessible to the Adabas load library being used by your nucleus job. Adabas first tries to load ADALIC and, if unsuccessful, it reads from a DDLIC data set referenced in ADANUC.
2. Update your Adabas ADANUC nucleus jobs to reference the license, as described in *Step 5: Customize and Start the Adabas Nucleus and Test Adabas Communications*.

Step 2: Allocate and Format the Adabas Database (SMA Job Number I030)

Customize and run the ADAFRM utility job to allocate and format the Adabas database. The following must be customized:

- Data set names for the database and libraries;

- Volumes for libraries and data sets for the database;
- Space allocation for data sets for the database;
- The Adabas SVC number, the database ID, and database device type(s);
- Sizes of the data sets for each ADAFRM statement.

Step 3: Define the Global Database Characteristics (SMA Job Number I030)

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

- Data set names of the database and libraries;
- The Adabas SVC number, the database ID, and database device type(s);
- ADADEF parameters.

Step 4: Load the Demonstration Files (SMA Job Number I050)

Customize and run the job:

- ADALODE to load the sample demo file EMPL;
- ADALODV to load the sample demo file VEHI;
- ADALODL to load the sample LOB file with LB option fields, LOBFILE;
- 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:

- Data set names for the database and libraries;
- The Adabas SVC number, the database ID, and database device type(s);
- ADALOD parameters.

Step 5: Customize and Start the Adabas Nucleus and Test Adabas Communications (SMA Job Number I040)

Customize and run an appropriate ADANUC job to start the Adabas nucleus. (This processing will also test basic Adabas communications.) The following modifications to the ADANUC job must be made:

1. Software AG licensing requires that the modules LICMAIN and LICUTIL be loaded when your nucleus starts up. These modules are distributed in the MLC*vrs*.LOAD library. You must either:

- Copy LICMAIN and LICUTIL into ADA*vrs*.LOAD; or
- Concatenate MLC*vrs*.LOAD with ADA*vrs*.LOAD. A sample job, ADANUCS, demonstrates how to do this.

2. Verify that the license file is correctly referenced in the ADANUC job. Do either of the following:

   - Verify that the ADALIC load module, installed in *Step 1: Check, Prepare, and Install the Product License File*, is stored in a load library that is accessible to the Adabas load library. Add the user load library in which ADALIC resides to the STEPLIB concatenation of the Adabas nucleus job or copy the ADALIC library into ADA*vrs*.LOAD.
   - Verify that there is no ADALIC load module accessible to the Adabas load library and that the following DD statement is included in the ADANUC job:

     ```
     //DDLIC     DD  DISP=SHR,DSN=dsn
     ```

     where *dsn* is the data set name of the license file loaded from the tape (in ASCII format). Note that *dsn* could reference a member in a partitioned data set.

     Note:
     Adabas first tries to load ADALIC and, if unsuccessful, it reads from the DDLIC data set.

3. Data set names for the database and libraries must be customized for your installation.

   Note:
   Be sure to include appropriate user load libraries.

4. The Adabas SVC number, the database ID, and database device type(s) must be customized for your installation.

5. ADARUN parameters must be customized for your installation.

Step 6: Test Adabas Address Space Communications, If Appropriate

Customize and run the job ADAREP in MULTI mode with the CPEXLIST parameter to test Adabas address space communications. The following must be customized:

- Data set names for the database and libraries;
- The Adabas SVC number, the database ID, and database device type(s);
- ADAREP parameters.

Step 7: Load Adabas Online System Add-On Product, If Appropriate (SMA Job Number I061)

Customize and run the job ADAINPL to load the Adabas Online System (AOS) into a Natural system file using the appropriate batch version of Natural (read the Adabas Online System documentation to determine its Natural requirements). The following items must be customized:

- Data set names of the database and libraries;

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

Alternatively, install the AOS demo version delivered with Adabas: see the section *Installing the AOS Demo Version*.

Step 8: Terminate the Adabas Nucleus

Communicate with the Adabas nucleus to terminate the session either with an ADAEND operator command using the OS Modify command:

```
F jobname,ADAEND
```

—or

```
P jobname
```

where *jobname* is the job or task name of the started nucleus.

Step 9. Back Up the Database

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

- Data set names of the database and libraries;
- The Adabas SVC number, the database ID, and device type(s);
- ADASAV parameters.

Step 10: Insert the ADARUN Defaults

The member DEFAULTS in the Adabas JCL library can be modified to set the ADARUN defaults. The following must be customized:

- Data set names of the database and libraries;
- ADARUN user defaults:
  - Device type(s) (default: 3390)
  - SVC number (default: 249)
  - Database ID (default: 1)

Customize and run the DEFAULTS job to set the ADARUN defaults using the OS ZAP utility.

Step 11: Install the Required TP Link Routines for Adabas

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

SVC Integrity Validation

In the past, the presence of multiple SVCs with the same subsystem ID has resulted in a single ID table being used by different SVCs. This has caused problems, some of them serious (abnormal nucleus termination or corruption of the database).

To eliminate this danger, the Version 8 SVC checks to ensure that the SVC accessing the ID table is the same as the one that was used by ADASIP/ADASIR to initialize the table. If the SVCs are not the same, an abend 650 occurs.

Abend 650 occurs when an incorrect SVC number is specified in the ADARUN parameters for a nucleus. It can occur during Adabas initialization, during the first Adabas call from a user program, or when the ID table is queried by another Software AG server such as Entire Net-Work.

ADASIP has been enhanced to prevent this from arising. If you attempt to install an instance of ADASVC using an SVC number that is presently associated with another subsystem name, ADASIP will terminate with abend 435.

Requirements for Cross-Memory Services

Due to the implementation of cross-memory services in z/OS, the following points should be noted when running an Adabas nucleus in MULTI mode:

- a maximum of one step of a job can establish the cross-memory environment. This means that a job can include at most one step that is a target (for example, an Adabas nucleus).
- cross-memory accesses may not be made to a swapped-out address space. Therefore, the address space of an Adabas nucleus is set to “nonswappable” for the duration of the nucleus session. This can increase the installation’s real storage requirements. This behavior is documented in the IBM manual *Extended Addressability Guide*, chapter Synchronous Cross-Memory Communication.
- when a nucleus with an active cross-memory environment terminates either normally or abnormally, the entire address space including any initiator is also terminated.

The ASID representing this address space is not reassigned until the next IPL. Therefore, you should choose a sufficiently high value for the MAXUSERS parameter in the active IEASYS*xx* member of SYS1.PARMLIB or—if your system supports it—the RSVNONR parameter in the same member can be adjusted accordingly. Also, the Adabas nucleus should not be stopped and started without good reason.

This is described in the manuals referred to in the topics Recovery Considerations and Resource Management. Additional information can be found in IBM APARs OZ61154, OZ61741, and OZ67637.

To make its services available to all address spaces in the system, the Adabas nucleus must obtain a system linkage index (LX) from z/OS. The LX is a reserved slot in the linkage space of all address spaces, and permits system-wide linkage to all address spaces.

If your configuration is using z/OS 1.6 or later and your hardware supports the Address Space and LX Reuse Facility (ALRF), the version 8 ADASVC will make use of reusable system LXs. Otherwise, a non-reusable system LX will be used as in previous releases. Reusable system LXs resolve the constraints imposed by non-reusable LXs. The remainder of this section discusses these constraints.

The number of non-reusable LXs set aside by z/OS for system use is rather small (usually 165 out of a possible 2048).

Because of the way z/OS use cross-memory services, non-reusable system LXs obtained by Adabas cannot be returned to z/OS until the next IPL. However, the system that owns the LXs can reuse them, even for a different address space. Adabas makes use of this feature by identifying used LXs in a table in CSA, where they are available to future nuclei.

The number of non-reusable system LXs can be specified in the member IEASYSxx contained in SYS1.PARMLIB, using the NSYSLX parameter. If you change this value, you must perform an IPL to make the change effective.

To determine an appropriate NSYSLX value, consider the following points:

- some LXs are probably already being used by other system functions. Therefore, the chances of creating an LX shortage for other users is small.
- Adabas requires one system LX for each Adabas nucleus (or any other target) that will be active concurrently. A value of decimal 64 would allow concurrent execution of up to 64 Adabas nuclei or other targets with little chance of restricting other components using LXs.
- Entire Net-Work Version 5 uses only one LX and one ID table entry, regardless of how many remote databases it must represent. This is unlike the pseudo-MPM concept of earlier Entire Net-Work versions.
- whenever ADASIP is executed with the REPLACE option, all LXs saved in the current ID table are lost until the next IPL.

Likewise, if a session ends either normally with the FORCE operator command or abnormally during ESTAE processing (for example, by an S222 operator cancel or by a S722 spool limit exceeded abend during a snap dump), the LX also cannot be recovered until the next IPL.

Any commands sent to these targets receive an S0D6 abend. Any attempt to restart the nucleus results in an ADAM98 message DUP ID (LOCAL), followed by an abend. To resolve both of these problems, restart the nucleus with the ADARUN FORCE=YES and IGNDIB=YES parameters.

The first target that tries to obtain a system LX when none is available ends with an S053 abend code and reason code 0112. No additional targets can be started until the next IPL.

Requirements for Global Resource Serialization

Adabas uses Global Resource Serialization (GRS) to synchronize the execution of Adabas nuclei and utilities at certain points in their processing. It is vital that GRS be set up correctly in the system so that GRS requests by Adabas will be effective.

When setting up GRS, consider the following:

- Adabas uses the GRS macros ENQ and DEQ with systems-wide scope (SCOPE=SYSTEMS) and major name ‘ADABAS’ (QNAME).
- if the database resides on disks that are shared between multiple images of the operating system (multiple LPARs or machines) and Adabas nuclei or utilities may be run against the database from several of these images, make sure that GRS is installed in a way that systems-wide ENQ requests are effective on all of these system images.

Using EXCPVR

Adabas performance and system throughput are improved when you use EXCPVR rather than EXCP, due to a reduction in channel program translation overhead.

For Adabas to invoke EXCPVR, the nucleus must be running with APF authorization. All Adabas modules must be in APF-authorized libraries, and all libraries in the JOBLIB or STEPLIB concatenations must be APF-authorized.

Running the Adabas nucleus and utilities APF-authorized is now a standard. For more information, read *APF Authorization Requirement*.

Prior versions of Adabas tied the use of EXCP or EXCPVR for database I/Os to the APF-authorization of the load library. EXCP was always used when running non-APF-authorized; EXCPVR was always used when running APF-authorized. If you wanted to use EXCP when running APF-authorized, you were required to apply special A$- or AY- zaps.

Adabas 8 introduces a new ADARUN parameter, EXCPVR, available in z/OS environments. Using this parameter, you can specify whether EXCP or EXCPVR is used when running APF-authorized. If the EXCPVR parameter is set to "YES", EXCPVR is used; if it is set to "NO", EXCP is used. If the EXCPVR parameter is set to "YES" when running from a non-APF-authorized load library, the EXCPVR parameter is ignored. For complete information on the use of this parameter, read *EXCPVR : Control EXCP or EXCPVR Use*.

In addition to the EXCPVR parameter, another ADARUN parameter, PGFIX, affects EXCPVR use. When EXCPVR is used on z/OS systems, the PGFIX parameter controls whether pages containing I/O control blocks are released after I/O processing is completed or after the job has ended. This parameter is valid only for z/OS users of EXCPVR. When the PGFIX parameter is set to NO, pages containing the I/O control blocks are fixed only for the duration of the I/O processing; when it is set to YES, pages containing the I/O control blocks are fixed for the duration of the job. For complete information on the use of this parameter, read *PGFIX: EXCPVR Page Fixing Control*.

Creating a Shareable ADARUN

The ADARUN module delivered in the Adabas load library is not reusable. If you need a shareable ADARUN, you will need to relink it with the REUS=YES link-edit attribute.

Linking ADARUN with the reusable option permits several programs running in the same address space to share the same ADARUN and ultimately, the same copy of ADALNK. This is important when it is necessary to have only one Adabas user ID for the different programs, and is also needed if single copies of ADALNK user exits are required.

To create a shareable ADARUN, use the sample job JCLLINRR in the JOBS library to relink it with the REUS attribute.

If both nonreusable and reusable versions of ADARUN are required, they must be located in different load libraries since both must be loadable using the name ADARUN.

Storage Above 16 MB

Adabas can acquire a number of its required areas, including buffer space, above the 16-MB addressing limit, allowing Adabas to increase the buffer pool size.

To reverse the space allocation to be below the 16-MB limit, set the AMODE value in the MODE statement in the example below to AMODE(24):

```
//LINKRUN EXEC PGM=IEWL,PARM='REUS'
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//ADALIB DD DSN=user.loadlib,DISP=SHR
//SYSLMOD DD DSN=user.loadlib,DISP=SHR <=APF-authorized library
//SYSLIN DD *
MODE AMODE(24),RMODE(24)
SETCODE AC(1)
INCLUDE ADALIB(ADARUN)
NAME ADARUN(R)
```

In addition, Adabas must be run with a sufficient REGION specification, either on the JOB or EXEC statement or as an installation default. For example:

```
//BIG JOB ...,REGION=30M,...
```

Storage Above 2 GB (64-Bit)

- Real Storage
- Virtual Storage

Real Storage

Adabas can exploit storage occupying real pages above the 2-gigabyte line. This capability allows Adabas I/Os to use 64-bit real addresses.

Support for 64-bit real storage is available whether you are running APF-authorized (using EXCPVR) or not (using EXCP). The run mode is indicated in the ADAI65 message:

```
ADAI65 EXCPVR IS {BEING | NOT BEING} USED FOR THIS RUN IN ESA64 MODE
```

Support for 64-bit real storage requires either

- z/OS R10 in ARCHLEVEL=2 (that is, z/architecture mode); or
- z/OS 1.2 or above

on a processor of the IBM 2064 family with an LPAR greater than 2 gigabytes for real storage allocation.

Virtual Storage

IBM supports 64-bit virtual storage only for z/OS 1.2 or above.

Software AG provides support for IBM's 64-bit virtual storage in Adabas. The ADARUN V64BIT parameter allows you to indicate whether the Adabas nucleus should use virtual storage above the 2 gigabyte bar. If your z/OS operating system supports 64-bit virtual storage, you can request that the Adabas nucleus use it also. In addition, the ADARUN LARGEPAGE parameter allows you to indicate whether the Adabas nucleus should use large pages (1 MB pages of real storage above the 2 gigabyte bar). If your z/OS operating system supports 64-bit virtual storage and large pages, you can request that the Adabas nucleus use large pages.

Software AG also provides support for 64-bit virtual storage in the product Adabas Caching Facility (ACF). Contact your Software AG representative for more information.

A demo of Adabas Caching Facility is delivered in the ADA*vrs*.ALLINPL file (where *vrs* is the number of the latest Adabas *version* delivered on the tape).

Applying Zaps

Use the z/OS AMASPZAP utility to apply zaps in the respective operating system; this method verifies (VER) and replaces (REP) data. The following sample JCL executes AMASPZAP:

```
//ADAZAP JOB
//STEP1 EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=X
//SYSLIB DD DSN=ADABAS.ADAvrs.LOAD,DISP=SHR
//SYSIN DD *
(zap control statements)
/*
//
```

—where the following are examples of zap control statements:

```
NAME membername csectname
VER displacement data
REP displacement data
IDRDATA (up to eight bytes of user data)
* (comment)
```

Note:
In VER and REP statements, spaces must be used to separate command, displacement, and data. Commas are acceptable data separators; however, commas with spaces or spaces alone are not, and may cause errors.

Adabas 8 Adalink Considerations

Note:
For information about connecting a database that is enabled for data conversion using the universal encoding service (UES), see the section *Enabling Universal Encoding Support (UES) for Your Adabas Nucleus*.

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

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

A pre-command user exit and a post-command user exit 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 (ADARSP216) 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 (ADARSP213)	No ID table
216 (ADARSP216)	LUEXIT1 suppressed the command
218 (ADARSP218)	No UB available

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 modules.

On the first Adabas call, ADAUSER 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.