Installation Procedure

This section describes the preparation and installation of Adabas on IBM z/VM systems.

To use the information in this section, you need to be familiar with z/VM and CP concepts such as directories and virtual machines. See the *Glossary* for an explanation of terms referenced.

Note:
See section *VM/GCS* for VM/GCS-related information to support some Software AG products such as Entire Net-Work. Adabas itself no longer supports VM/GCS.

- Installation Checklist
- Contents of the Release Tape
- Installation Steps
- z/VM Operating System Environment
- Adabas File Support under z/VM
- Entering Operator Commands
- Setting Defaults for ADARUN
- Installing New SM Levels
- Applying Zaps in z/VM
- IUCV Security Options
- Adabas 8 ADALNK/ADAUSER Installation Considerations
- Adabas 7 ADALNK/ADAUSER Installation Considerations

Installation Checklist

The following list provides an overview of the Adabas installation procedure on a z/VM system.

Step	Description	Additional Information
1	Define virtual machines for the Adabas environment.	Two virtual machines are required for operating Adabas. At least two more virtual machines are recommended: • Adabas nucleus (required) • ID table manager (required) • database administrator (recommended) • user programs (recommended) For specific information on the CP directory requirements for each of these virtual machines, refer to the individual sections describing the virtual machines.
2	Define minidisk space for the Adabas library disk.	If a DBA virtual machine has been defined, the library minidisk should be defined by an MDISK statement in the CP directory for the DBA virtual machine; otherwise, it should be defined in the CP directory of the Adabas nucleus machine. For detailed information about the Adabas library minidisk see the section *Adabas Library Requirements*.
3	Define minidisk space for the Adabas database.	Some effort should be made to place these minidisks on different channels, both virtual and real. For detailed information about the Adabas database minidisks, see the section *Disk Space Requirements for the Database*.
4	Load the Adabas release tape and subsequently install a database.	The procedure for doing this is described in *Installation Steps*.

Note:
If you plan to use the Adabas Recovery Aid (ADARAI) utility, specific changes must first be made to the PROFILE and ADF*dbid* EXECs.

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/VM environments.

Library Name	Description
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*.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.
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*.TAPE	The library for Adabas. The *vrs* in the library name represents the *version* of Adabas.
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*.L014	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.

Library Name	Description
APS*vrs*.TAPE	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*.TAPE	The 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*.TAPE	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.

Installation Steps

The following steps are required to load the release tape and install Adabas.

To load the release tape and install Adabas:

1. LOGON to and DISCONNECT from the ID table manager virtual machine.
2. LOGON to the Adabas nucleus virtual machine. If a database administrator (DBA) virtual machine has been defined, DISCONNECT from Adabas; otherwise, ATTACH a tape drive to Adabas as unit 181 and continue with step 4.
3. LOGON to the DBA virtual machine and ATTACH a tape drive as drive 181.
4. ACCESS the Adabas library disk with a filemode other than "A". Filemode "C" is recommended.
5. Issue command TAPE FSF 4.
6. Issue command TAPE LOAD * * *fm*.
7. Create a PROFILE EXEC for the DBA virtual machine.

   Refer to the section *z/VM Operating System Environment* for an example of the CP directory entries for the DBA virtual machine.

   Include the following functions in the PROFILE EXEC:

   - MULTI write LINK commands to the Adabas database minidisks (ASSO, DATA, WORK, TEMP, SORT).
   - ACCESS for the library disk as a read only extension of the A-disk (ACC *cuu fm*/A).
   - Issue EXEC SETTXTLB.
   - Define ADARUN, DATADEF, DISPDD, and RELDD as nucleus extensions. This can be done by executing the NUCXTNTS EXEC with no parameters.

8. Execute the PROFILE EXEC.
9. Create a zap file for ADAITM if you are going to modify the default values for the target-ID, VMID, or restart parameters:
   - target ID of ADAITM at location X'20'. (default X'FFFF').
   - VMID of the DBA virtual machine at location X'28'. (default: 8 blanks)
   - automatic restart parameter at location X'30' (default: C'Y').
10. Create a zap file for ADALDI if you are going to modify the default values for the VMID of the ID table manager.

    The VMID of the ID table manager is at location X'34' (default C'DBIDSERV').

    Apply the zaps if necessary.
11. Issue the following commands, where *nnnnn* is the database ID and *fm* is the filemode. Modify the copied EXECs, which always have a filename containing the five-digit DBID (with leading zeros, if needed), as follows:

```
COPY ADADEFS EXEC fm ADFnnnnn EXEC fm
COPY ADAFRM CONTROL fm FRMnnnnn CONTROL fm
COPY ADADEF CONTROL fm DEFnnnnn CONTROL fm
COPY RUNDB CONTROL fm RDBnnnnn CONTROL fm
COPY RUNDEV CONTROL fm RDVnnnnn CONTROL fm
```

12. Copy the ADALOD control statements to load the Employees, Vehicle and Miscellaneous demo files:

```
COPY loadname CONTROL fm demoname LODLIB fm
```

    —where *loadname* is EMP, VEHILOD or MISC, and *demoname* is EMPLLOD, VEHILOD, and MISCLOD, respectively. Change the corresponding LODLIBs as required.

    Note:
    The Personnel and large object (LOB) demo files are delivered with matching ADALOD input files, so no CMS COPY command is required for these demo files.
13. Create the file DB*nnnnn* VOLUMES *filemode*, where *nnnnn* is the five-digit database ID.

    This file must contain one record per direct access mini-disk (that is, one for the Associator, one for Data Storage, and so on). Each record must contain the file name, file type, disk label, and virtual unit address (in that order) where the disk will be formatted and reserved. The record entries must be separated by blanks.

    Copy the ADALOD control statements to load the demo files.
14. Execute the DBINIT EXEC specifying the DBID as a parameter.
15. Execute the ADALOD EXEC specifying FILENAME (EMPL, ADAPERL, ADAPERS, VEHI, MISC) and the DBID as parameters.

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

16. LOGON to the ID table manager virtual machine.
17. Create a PROFILE EXEC for the ID table manager virtual machine.

    Refer to the section *z/VM Operating System Environment* for an example of the CP directory entries for the ID table manager virtual machine.

    Include the following functions in the PROFILE EXEC:

    - multi-read LINK commands to the Adabas library disk (ADAV*vvv* TXTLIB).
    - ACCESS for the library disk as a read only extension of the A-disk (ACC *cuu fm*/A);
    - EXEC SETTXTLB;
    - LOAD ADAITM * (START.
18. Execute the PROFILE EXEC created in step 17.
19. DISCONNECT from the ID table manager virtual machine.
20. LOGON to the Adabas virtual machine.
21. Create a PROFILE EXEC for the Adabas virtual machine.

    Refer to the section *z/VM Operating System Environment* for an example of the CP directory entries for the Adabas virtual machine.

    Include the following functions in the PROFILE EXEC:

    - multi-read LINK commands to the Adabas library disk (ADAV*vvv* TXTLIB).
    - ACCESS for the library disk as a read-only extension of the A-disk (ACC *cuu fm*/A).
    - EXEC SETTXTLB
    - * SET STORECLR ENDCMD
    - EXEC NUCXTNTS (defines ADARUN, DATADEF, DISPDD, and RELDD as nucleus extensions).
22. Execute the PROFILE EXEC created in step 21.
23. Prepare and install the product license file.

    The product license file is supplied on the individual customer installation tape or separately via an e-mail attachment. To install it you must:

- Transfer the license file from tape or e-mail to your A disk as LICENSE DATA. Instructions for performing these transfers are provided in *Transferring a License File to a z/VM Host*. Be sure to specify a file name of "LICENSE" and a file type of "DATA" during the transfer.
- Once the license file is transferred , you can install it. Instructions for installing the license are provided in this step.

Installing the license file

To install the product license file, complete the following steps:

1. Verify that the license file is located on the A disk as LICENSE DATA (with RECFM=F and LRECL=80), taking care to preserve its format as ASCII.
2. Verify that the contents of the MLC*vrs*.TAPE library have been made available to the Adabas library disk.
3. Run the LICMAKE EXEC. This EXEC invokes the LICUTIL utility to generate file LICENSE ASSEMBLE A.
4. Run the ASMLICAM EXEC. This EXEC assembles LICENSE ASSEMBLE, creating ADALIC TEXT. This file must be available to the Adabas nucleus.

Note:
Two additional EXECs are provided to assist with licensing Adabas in VM. The LICCHECK EXEC invokes the LICUTIL utility to separately validate the license file LICENSE DATA; the output is written to file LICENSE LISTING. The LICMDATA EXEC invokes the LICUTIL utility to list the current machine data information; the output is also written to file LICENSE LISTING.

24. Start the Adabas nucleus by entering ADANUC, specifying the DBID as a parameter.
25. If a DBA virtual machine has been defined, DISCONNECT from the nucleus virtual machine before continuing; otherwise, continue with step 25.
26. LOGON to the database administrator (DBA) virtual machine.
27. Execute the ADAREP EXEC, specifying as parameters the DBID and the filename of the z/VM file containing the ADAREP control statements (i.e. REPCPLST).
28. Stop the Adabas nucleus by entering ADAEND from the console of the Adabas virtual machine, or from a secondary console.
29. Execute the ADASAV EXEC specifying the filename of the z/VM file that contains the ADASAV control statements (i.e. SAVE) and the DBID as parameters.

    For systems using the Adabas Online System, perform the following additional step:

30. LOGON to the DBA virtual machine.

This completes the installation.

z/VM Operating System Environment

In z/VM, the following entities are required or recommended to run Adabas.

- Adabas nucleus (required)
- ID table manager (required)
- database administrator (recommended)
- user programs (recommended)

Each of the above Adabas entities operates as a task within its own z/VM environment, known as a virtual machine. Each virtual machine comprises system resources that emulate virtual storage, virtual DASD (minidisk) space, and I/O capabilities such as an operator console and printer.

The actual system resource allocated to each Adabas virtual machine is either predefined in a CP directory or dynamically defined in the PROFILE EXEC that is invoked when each Adabas virtual machine begins operating. The following sections describe each of the Adabas virtual machines and how its resources are initially defined when installing Adabas.

Following this description is specific information concerning individual virtual machine requirements and operations in a z/VM environment.

- Adabas Nucleus Virtual Machine
- ID Table Manager Virtual Machine
- Database Administrator (DBA) Virtual Machine
- User Virtual Machine
- Adabas Library Requirements
- Disk Space Requirements for the Database

Adabas Nucleus Virtual Machine

The Adabas nucleus executes in its own virtual machine, which normally runs disconnected. The nucleus virtual machine requires various z/VM minidisks. The typical required virtual storage size for the Adabas nucleus machine is 32 MB or larger.

- Allocating Adabas Minidisk Space
- Communicating with Other Virtual Machines
- Nucleus Extension Requirements
- Providing DBA Control of the Nucleus
- Nucleus Directory

Allocating Adabas Minidisk Space

An A-Disk of at least 500 4-kilobyte blocks is required, and if no DBA virtual machine exists, either a library minidisk of at least 1500 4-kilobyte blocks is needed, or a read only LINK to another nucleus virtual machine containing the library minidisk. Refer to the section *Adabas Library Requirements*.

Typically, the database resides on z/VM minidisks defined in the CP directory of the Adabas nucleus virtual machine. These minidisks must have multi-write passwords in the CP directory. See the section *Disk Space Requirements for the Database*.

Communicating with Other Virtual Machines

To authorize the nucleus' use of z/VM IUCV, place the following statement in the CP directory:

```
IUCV ALLOW (PRIORITY)
```

—where "PRIORITY" is optional.

If access to the nucleus machine is to be restricted, the IUCV statement should be in the CP directory of the user machines. In addition, the CP directory must have the MAXCONN parameter of the OPTION statement set high enough to accommodate one path to each z/VM or guest operating system's user machine and two paths to the ID table manager virtual machine. For example:

```
OPTION MAXCONN 5
```

—would be sufficient to support three Adabas user machines and the ID table manager machine.

Before the Adabas nucleus is started, the following commands must also be issued:

```
ACCESS cuu fm/A
EXEC SETTXTLB
SET STORECLR ENDCMD
—where
cuu is the virtual unit address of the library minidisk.
fm is the  z/VM filemode.
```

If used, the online installation procedure creates these statements in the nucleus user machine's PROFILE EXEC file.

Nucleus Extension Requirements

The Adabas nucleus machine requires four nucleus extensions with the following attributes:

```
ADARUN USER SERVICE
DATADEF SYSTEM
DISPDD SYSTEM
RELDD SYSTEM
```

The NUCXTNTS EXEC may be used to load the four nucleus extensions. If the online installation procedure is used, it adds a statement in the PROFILE EXEC to invoke the NUCXTNTS EXEC when the nucleus virtual machine starts.

If a new version of one of the nucleus extensions is to be activated the following commands must be entered:

```
ERASE extname MODULE A
NUCXDROP extname
NUCXTNTS
```

—where *extname* is the nucleus extension name. If NUCXTNTS is invoked without the two previous commands, the old version of the program remains active.

Providing DBA Control of the Nucleus

The DBA virtual machine can be defined to z/VM as a secondary console for the Adabas nucleus machine. This allows the DBA to control the nucleus machine, and issue Adabas operator commands using the SEND command from a terminal. The secondary console support is defined by specifying the CONSOLE statement in the CP directory of the Adabas nucleus machine as follows:

```
CONSOLE 009 devtype T dbavmid
```

where *devtype* is the DBA console device type and*dbavmid* is the DBA virtual machine ID.

Nucleus Directory

The following is an example of the entries in the CP directory for the Adabas nucleus virtual machine running in ESA mode:

```
USER ADA00001 ADA00001 6MB 64M G
ACCOUNT xx xxxxx
IPL CMS PARM AUTOCR
MACHINE ESA
OPTION MAXCONN 100
IUCV ALLOW PRIORITY
CONSOLE 009 3215 T SAGDBA
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
```

The LINK and MDISK statements are as follows:

```
LINK MAINT 190 190 RR
LINK MAINT 19E 19E RR
.
.
MDISK 191 3380 067 005 vvvvvv MR rpassword wpassword mpassword
MDISK 200 3380 304 016 vvvvvv MW rpassword wpassword mpassword
MDISK 300 3380 400 051 vvvvvv MW rpassword wpassword mpassword
MDISK 400 3380 451 021 vvvvvv MW rpassword wpassword mpassword
MDISK 410 3380 472 011 vvvvvv MW rpassword wpassword mpassword
MDISK 420 3380 483 006 vvvvvv MW rpassword wpassword mpassword
MDISK 430 3380 489 002 vvvvvv MW rpassword wpassword mpassword
MDISK 431 3380 491 002 vvvvvv MW rpassword wpassword mpassword
MDISK 440 3380 493 002 vvvvvv MW rpassword wpassword mpassword
MDISK 441 3380 495 002 vvvvvv MW rpassword wpassword mpassword
```

The ECMODE OPTION statement, though not required, is recommended because it enables the use of certain VM assists.

ID Table Manager Virtual Machine

The ID table manager executes in its own virtual machine, which normally runs disconnected. For information about the function of the ID table manager, refer to the section *Functions of the ID Table Manager*.

The virtual storage size required for the ID table manager is four megabytes. The A-disk must be at least 100 4-kilobyte blocks and have a read-only link to the Adabas library disk (where the ADAV*vrs* TXTLIB and all CONTROL files reside).

By default, the ID table manager allows 20 ID table entries. A zap can be used to extend this limit:

```
NAME ADAITM ADAITM
VER 0022 00FF     dafault 20
REP 0022 00xx      new default
```

- Communicating with Other Virtual Machines
- ID Table Manager Directory
- Functions of the ID Table Manager

Communicating with Other Virtual Machines

The ID table manager must be authorized for IUCV communication. The following entries should be placed in the CP directory:

```
IUCV ALLOW (PRIORITY)
IUCV ANY (PRIORITY)
```

—where “PRIORITY” is optional, but recommended. In addition, the OPTION statement’s MAXCONN parameter in the CP directory must be set high enough to accommodate an IUCV path to each user virtual machine, either z/VM or a guest operating system, and two paths to each Adabas nucleus machine.

The PROFILE EXEC of the ID table manager virtual machine should contain the following commands:

```
LINK SAGDBA cuu cuu RR rpassword
ACC cuu fm/A
EXEC SETTXTLB
LOAD ADAITM (RESET ADAITM
START * ...
```

—where *cuu* is the virtual unit address of the library minidisk and *fm* is the z/VM filemode.

Software AG recommends starting the ID table manager machine automatically at system startup time using the AUTOLOG command in the PROFILE EXEC of the AUTOLOG1 virtual machine. If set up to do so, the ID table manager virtual machine’s PROFILE EXEC executes AUTOLOG to start the Adabas virtual machines.

The ID table manager accepts parameters for the following values:

- DBA virtual machine ID, keyword DBAVMID;
- ID table manager target ID, keyword NODEID;

- ID table manager node name, keyword NODENAME;
- Entire Net-work IUCV resource ID, keyword NETRESID.

These keyword parameters are specified in the z/VM START command. For example, to execute the ID table manager with a node ID of 1001 and a node name of CMSNODE, enter the following START command:

```
START * NODEID 1001 NODENAME CMSNODE
```

Except for the DBAVMID parameter, these parameters are only relevant to operation with Entire Net-Work.

ID Table Manager Directory

The following is an example of the entries in the CP directory for the ID table manager virtual machine running in ESA mode:

```
USER DBIDSERV DBIDSERV 6M 8M G
ACCOUNT xxxxxxxx
OPTION MAXCONN 100
IUCV ALLOW PRIORITY
IUCV ANY PRIORITY
IPL CMS ESA PARM AUTOCR
CONSOLE 009 3215 T SAGDBA
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
LINK MAINT 190 190 RR
LINK MAINT 19E 19E RR
MDISK 191 3380 2741 vvvvvv MR rpassword
```

Functions of the ID Table Manager

Under z/VM, commonly addressable and modifiable storage areas are not available. This means that there can be no centrally located routing table as in other operating systems.

Instead, this information is maintained and distributed by the ID table manager program, ADAITM, which executes in its own virtual machine. It is a required virtual machine and operates continuously in disconnected or background mode.

ADAITM also provides all information needed for communication between z/VM and any guest operating systems running under VM.

The Adabas nucleus (ADAMPM) and ADALNK establish an IUCV communications path to the virtual machine running ADAITM during the startup procedure. The VMID of the ID table manager virtual machine is at a fixed location in ADALDI, a program used by ADAMPM and ADALNK for communications. Both this VMID and the target ID of the ID table manager can be set at installation time and should not be changed. The default VMID is DBIDSERV and the default target ID is 65535.

The DBA virtual machine ID, ID table manager target ID, and node name can all be specified in the ID table manager PROFILE EXEC as parameters:

```
LOAD ADAITM ( RESET ADAITM'
START * DBAVMID vm-id NODEID node-id NODENAME node-name
NETRESID res-id
```

where *vm-id* is the ID of the DBA virtual machine, *node-id* is the ID table manager target ID, *node-name* is the ID table manager node name, and *res-id* is the Entire Net-Work IUCV resource ID.

If desired, the DBA virtual machine ID and ID table manager target ID (node ID) can also be zapped.

With the exception of the DBAVMID parameter, the parameters above are only relevant to operation with Entire Net-Work.

The following zap modifies the default DBID and the database administrator machine VMID (described later) in the ID table manager:

```
NAME ADAITM ADAITM
VER 0020 FFFF
VER 0028 4040,4040,4040,4040
REP 0020 FFFF
REP 0028 E2C1,C7C4,C2C1,4040
```

This zap illustrates how to modify VMID of the ID table manager in ADALDI:

```
NAME ADALDI ADALDI
VER 0034 C4C2,C9C4,E2C5,D9E5
REP 0034 C4C2,C9C4,E2C5,D9E5
```

If an ADAITM program error occurs and the restart option is active, all information concerning the Adabas z/VM environment is written to the z/VM file ADAITM RESTART on the ID table manager's A-Disk and a message is sent to both the CP operator and (if one has been defined) to the database administrator virtual machine consoles. ADAITM then severs all IUCV paths, places a command in the z/VM program stack to reload and restart itself, and stops.

If the restart option is not active, no ADAITM RESTART file is written; the ID table manager logs off, and terminates all active nuclei abnormally.

During the time that ADAITM is not available, existing local communication paths may still be used; however, no new paths can be initiated and no new Adabas nuclei can be started. All remote communications are terminated.

After restarting, ADAITM reconnects to all Adabas z/VM nuclei and to any Adabas users in z/VM machines who are still active. If another error occurs during error recovery or restart, ADAITM sends a message to both the CP operator and the database administrator virtual machine consoles, and then logs off the ADAITM virtual machine. This causes all local Adabas nuclei to abend. Stopping the ADAITM virtual machine with FORCE also abends all local nuclei.

ADAITM accepts operator commands for administration purposes. These operator commands comprise two general categories: listing commands and trace commands. Listing commands (LIST*xxxx*) display console lists of the requested information, usually as AITM*nn* messages. Trace commands provide chronological information such as nucleus/user initialization and termination.

The commands are

Command	Description
DISPON	Displays events on the z/VM console as they occur
DISPOFF	Stops event display on the z/VM console
LISTLINK	Displays all active links to directly addressable network nodes (for Entire Net-Work systems only)
LISTLOG	Displays the contents of the logging area on the console
LISTMSG	Displays the contents of the NETITM MSGS file, which contains messages related to Entire Net-Work communication
LISTNODE	Displays all the active network nodes on the console
LISTTARG	Lists active targets (active nuclei, etc.) on the console
LISTUSER	Lists all user virtual machines on the console
LOGON	Records events in a logging area
LOGOFF	Stops logging of events
RESET	Clears the event logging area

Database Administrator (DBA) Virtual Machine

To give the database administrator (DBA) maximum control of the Adabas environment, a separate DBA virtual machine should be allocated. This allows the DBA to

- perform multi-user utility operations on multiple databases;
- maintain a single Adabas library disk, ADAV*vrs* TXTLIB; and
- issue operator commands for nuclei running in disconnected machines (when the DBA virtual machine is defined as a secondary console for the nucleus machine).

The DBA virtual machine requires a minimum of 4 megabytes of virtual storage. Depending on the Adabas utilities to be run in the DBA virtual machine and the parameters specified, more storage may be required. The standard virtual storage size is 4 megabytes.

Allocating DBA Minidisk Space

The A-Disk must be at least 750 4K blocks (or the equivalent). If this virtual machine is being used, the Adabas library disk must be defined in the CP directory for the DBA virtual machine. See the section *Adabas Library Requirements*.

Multiple write links must be defined from the DBA virtual machine to each database minidisk that the DBA machine supports. If the online installation procedure is used and LINK passwords are specified, LINK statements are automatically created in the DBA virtual machine’s PROFILE EXEC.

Communicating with Other Virtual Machines

The DBA virtual machine must be authorized for IUCV communication. The following entry should be placed in the CP directory:

```
IUCV ALLOW PRIORITY
```

The MAXCONN parameter of the OPTION statement in the CP directory must be set high enough to accommodate an IUCV path to the ID table manager and paths to each Adabas nucleus machine.

To execute Adabas utilities, the DBA virtual machine must have the following statements in its PROFILE EXEC:

```
ACCESS cuu fm/A
EXEC SETTXTLB
SET STORECLR ENDCMD
```

where *cuu* is the virtual unit address of the library minidisk and *fm* is the filemode of the library minidisk.

If used, the online installation procedure creates these statements in the PROFILE EXEC.

DBA Nucleus Extension Requirements

The database administrator virtual machine requires four nucleus extensions with the following attributes:

```
ADARUN USER SERVICE
DATADEF SYSTEM
DISPDD SYSTEM
RELDD SYSTEM
```

The NUCXTNTS EXEC may be used to load the four nucleus extensions. If the online installation procedure is used, it adds a statement to the PROFILE EXEC to invoke the NUCXTNTS EXEC when the DBA virtual machine starts.

If a new version of one of the nucleus extensions is to be activated, enter the following commands with NUCXTNTS:

```
ERASE extname MODULE A
NUCXDROP extname
NUCXTNTS
```

—where *extname* is the nucleus extension name.

If NUCXTNTS is invoked without ERASE and NUCXDROP, the old version of the program remains active.

DBA Directory

The DBA virtual machine's CP directory requires an OPTION statement. The following is an example of the entries in the CP directory for the DBA virtual machine:

```
USER SAGDBA SAGDBA 6M 8M G
ACCOUNT xxxxxxx
OPTION MAXCONN 10
IUCV ANY
IPL CMS PARM AUTOCR
```

User Virtual Machine

Each user virtual machine requires 1536 KB of virtual storage if Natural is to be used without a DCSS. If Natural is installed as a DCSS, 768 KB of virtual storage is sufficient.

- Allocating User Minidisk Space
- Communicating with Other Virtual Machines
- User Nucleus Extension Requirements
- User Directory
- ADARUN Control of User Programs

Allocating User Minidisk Space

Each user virtual machine must LINK and ACCESS to either the Adabas library disk (that is, where ADAV*vrs* TXTLIB resides), or to a sublibrary disk. Application programs executing in SINGLE user mode must use the Adabas library disk ADAV*vrs*. A sublibrary disk is created when users are to be isolated from the Adabas library disk. For more user information on the sublibrary disk refer to the section *Creating a User Sublibrary*.

Communicating with Other Virtual Machines

The user machine only needs to be authorized to use IUCV when the ID table manager and nucleus virtual machines do not have IUCV ALLOW statements in their respective CP directory entries. If neither the ID table manager nor the Adabas nucleus virtual machine has the IUCV ALLOW statements, issue the following statements in the user CP directory:

```
IUCV idtmvmid
IUCV nucvmid
```

where *idtmvmid* is the ID table manager ID and *nucvmid* is the Adabas nucleus virtual machine ID.

Refer to the section *IUCV Security Options*. The MAXCONN parameter of the OPTION statement in the CP directory must be set high enough to accommodate an IUCV path to the ID table manager and one path to each Adabas nucleus machine.

To execute ADALNK, the user virtual machine must have the following statements in its PROFILE EXEC:

```
ACCESS cuu fm/A
GLOBAL TXTLIB libname
```

where *cuu* is the virtual unit address of the library or sublibrary minidisk, *fm* is the filemode of the library or sublibrary minidisk, and *libname* is the TXTLIB filename.

User Nucleus Extension Requirements

A user virtual machine requires four nucleus extensions with the following attributes:

```
ADARUN SYSTEM SERVICE
DATADEF SYSTEM
DISPDD SYSTEM
RELDD SYSTEM
```

The NUCXTNTS EXEC must be issued with the parameter USER to load the four nucleus extensions.

To activate a new version of a nucleus extension, enter the following commands:

```
ERASE extname MODULE A
NUCXDROP extname
NUCXTNTS USER
```

—where *extname* is the nucleus extension name.

If NUCXTNTS is invoked without ERASE and NUCXDROP, the old version of the NUCXTNTS program remains active.

User Directory

The DBA virtual machine's CP directory requires an OPTION statement as shown in the following example:

```
USER ADAUSER1 ADAUSER1 6M 8M G
ACCOUNT xxxxxxx
IPL CMS PARM AUTOCR
MACHINE ESA
OPTION MAXCONN 10
CONSOLE 009 3215
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
LINK MAINT 190 190 RR
LINK MAINT 19E 19E RR
MDISK 191 3380 381 007 vvvvvv MR rpassword
```

ADARUN Control of User Programs

ADARUN control statements can be used to control user programs by issuing the appropriate DATADEF statement for DDCARD and optionally for DDPRINT. The DATADEF statement assures that all ADALNK errors are both displayed on the virtual console and recorded in the DDPRINT file.

Releasing the User Virtual Machine's Communication Environment

The Adabas communications environment can be reset and all storage allocated to that environment in a user virtual machine can be released by issuing the following z/VM command:

```
NUCXDROP ADARUN
```

Because the initialization and termination of the communication environment uses a lot of system resource, the NUCXDROP command is recommended only when no further use of Adabas communication from the user virtual machine is planned for the session.

Adabas Library Requirements

The Adabas library disk must be at least 5300 4K blocks (or about 30 cylinders). Each added database after the first adds a requirement of 100 4K blocks to the library disk. The library minidisk must be defined with a multiread or ALL password in either the DBA virtual machine's CP directory, or in the nucleus virtual machine's CP directory if no DBA machine has been defined. All virtual machines accessing the database must have read-only LINKs to the library minidisk.

Installations wishing to isolate multiuser application programs from the Adabas library disk can define a sublibrary on a commonly accessible minidisk, as described in the following section *Creating a User Sublibrary*. Application programs running in SINGLE user mode must use the Adabas library disk.

- Creating a User Sublibrary

Creating a User Sublibrary

A sublibrary disk is created to isolate users from the Adabas library disk. The sublibrary disk must be at least 200 4K blocks, be located on a commonly accessible minidisk, and have a multiple-read password. The sublibrary TXTLIB must contain the following members:

```
$SAGIOS ADALNK NETPARS NIUDEP
ADAIOR ADARUN NETRQM NIUEXT
ADAIOS ADAUSER (ADABAS) NETTQM
ADAILD NETBPM NETTRC
ADALDI NETIUCV NETTRT
```

To copy these members from the Adabas library disk, issue the following commands for each member:

```
FILEDEF IN DISK ADAVvrv TXTLIB fm (MEMBER membname
FILEDEF OUT DISK membname TEXT fmwork
MOVEFILE IN OUT
```

where *fm* is the filemode of the Adabas library disk, *fmwork* is the filemode of a work minidisk, and *membname* is the name of the member to be copied to the sublibrary.

Note:
ADAUSER has the member name ADABAS.

To create the new TXTLIB sublibrary, enter the following command:

```
TXTLIB GEN sublib membname1,membname2,....membnamex
```

where *sublib* is the name of the sublibrary and*membname1 — membnamex* are the names of members taken from the ADAV*vrs* TXTLIB.

After TXTLIB has been created, the text files can be deleted from the work minidisk. It is important to remember that a sublibrary needs separate maintenance; this means that zaps and new SM levels must be applied to both the Adabas library and each sublibrary. The NUCXTNTS, DEFNUCX, and SETXTLB EXEC routines must also be copied to the new sublibrary.

The following additional modules must be generated and then copied to the TXTLIB sublibrary minidisk:

```
ADAIOR DATADEF
ADALNK DISPDD
ADARUN RELDD
```

The z/VM module files having the same names should also be copied. You must also ensure that the sublibrary minidisk is updated whenever the main library minidisk is changed.

Each user machine utilizing the sublibrary minidisk must have the appropriate LINK, ACCESS and GLOBAL TXTLIB statements active. See the section *User Virtual Machine*.

Disk Space Requirements for the Database

The database can be on either DASD having z/OS or VSE VTOCs, or on VM minidisks. For z/OS or VSE disks, there must be multi-write LINKs or ATTACH commands to those disks in the PROFILE or CP directory of the nucleus virtual machine. For VM minidisks, the nucleus virtual machine's CP directory must contain the MDISK commands, and multi-write passwords must be defined.

The Adabas database requires at least five VM minidisks, as follows:

File	Suggested Unit Address	Required Space in Blocks	3380 Cylinders	3350 Cylinders
ASSO	200	8640 (2k)	30 + 2	35 + 1
DATA	300	10800 (4k)	70 + 2	85 + 1
WORK	400	1650 (4k)	10 + 2	15 + 1
TEMP	410	2250 (4k)	15 + 1	20 + 1
SORT	420	2250 (4k)	15 + 1	20 + 1

Optional minidisks and their recommended sizes and requirements are:

File	Suggested Unit Address	Required Space in Blocks	3380 Cylinders	3350 Cylinders
CLOG1	430	300 (4k)	1 + 1	2 + 1
CLOG2	431	300 (4k)	1 + 1	2 + 1
PLOG1	440	300 (4k)	1 + 1	2 + 1
PLOG2	441	300 (4k)	1 + 1	2 + 1

All sizes specified for the minidisks must be rounded up to the next full multiple of cylinders. For FBA devices, see the section *FBA Devices*.

The first cylinder or FBA pseudo-cylinder of each database minidisk is reserved for the z/VM directory. Therefore, one cylinder must be added to each minidisk allocated in the CP directory. The last column of each of the tables above accounts for the directory cylinder by adding one to the normal cylinder/pseudo-cylinder count.

The z/VM RESERVE command must be specified for each of the database minidisks. If the online installation procedure is being used, this RESERVE operation is performed when the installation procedure is executed.

Any Adabas virtual machine that executes multiuser utilities must define multi-write LINKs to the database minidisks.

Adabas File Support under z/VM

- DASD Supported by Adabas
- Sequential File Support
- Using Improved Data Recording Facility (IDRC) Tapes
- Tape Management with User Exit CMSUX1
- DATADEF File Assignments
- DATADEF Error Codes

DASD Supported by Adabas

Adabas z/VM supports direct access database files on z/VM-formatted disks or on CKD and FBA disks with a z/OS or VSE VTOC. The disks can be minidisks or real disks. The files may be contained on one or more real or virtual volumes.

z/VM-formatted disks must not be attached with the CP ATTACH command, but must be defined in the directory of the virtual machine or linked to it using the CP LINK command. Disks with database files need not be accessed using the z/VM ACCESS command. If a file spans multiple volumes, all of the devices must be of the same type and must have the same format; that is, either all devices must have a z/OS or VSE VTOC, or all must be z/VM-formatted.

Sequential File Support

Sequential files are supported as

- terminal files (input/output)
- card reader files (input only)
- card punch files (output only)
- printer files (output only)
- tape files (input/output)
- z/VM-formatted disk files (input/output)
- z/OS- or VSE-formatted disk files (input only)

The file format for terminal files must be fixed length with a maximum record length of 80. When an input file is assigned to terminal, the user is prompted with the name of the file being read. The end of a terminal input file is signaled by pressing ENTER without any input.

Sequential files on other unit record devices must also be fixed length. The maximum record length is the length supported by z/VM on the device.

All record formats and lengths are supported for tape files. If the tape file was created under an operating system which does not create HDR 2 records, the user must supply file format and length information. Tape files must reside on standard labeled volumes. The z/VM TAPE command may be used to initialize a tape volume and set its density. To avoid confusion during tape handling, it is highly recommended that each tape be labeled with a unique 6-byte name. Both multi-file volumes and multi-volume files are fully supported.

Sequential files on z/VM-formatted disks may be either fixed or variable length. Blocking is done by the z/VM file service routines. Unlike DASD volumes used for direct access files, disks containing sequential files must be ACCESSed. The z/VM file services do not differentiate between nonexistent files and empty files. To prevent open errors in Adabas utilities when a non-existent input file is used, a one-byte record containing a dollar sign ($) is written to sequential files on z/VM formatted disks at close time if no records have been written.

All record formats and lengths supported by z/OS or VSE may be used on sequential z/OS or VSE DASD files. If there is no record information in the label records (VSE), record format and length must be specified by the user. z/OS or VSE DASDs may not be used for sequential output files.

Adabas supports concatenation of sequential files if the record format and the logical record lengths of the concatenated files are the same. Concatenation of files on different storage media is also possible.

Using Improved Data Recording Facility (IDRC) Tapes

Sequential files to be read backward can be on either tape or z/VM-formatted disks. Exceptions to this are tape files created using the improved data recording capability (IDRC) on certain cassette units. When this facility is in use on certain other models of tape units, the read backward channel command is not supported. This point should be considered when planning recovery strategies for rebuilding the database after a possible failure.

Where supported, sequential files to be read backward may also be concatenated. If they are concatenated, they must be specified in the same order as though they were being processed normally.

The tape units to be used must be attached before file assignment by DATADEF unless they are to be dynamically allocated at open time by a tape management system. In this last case, a generic name can be specified in the DATADEF statement (see the section *DATADEF File Assignments*).

Tape units can be attached to any virtual address, and need not have any logical name. During the open process, Adabas reads the first block of the tape to verify that the correct volume is mounted. If the correct volume is not mounted or is not correctly initialized, the message ADAI48 is issued and the tape is unloaded. Message ADAI40 is then issued to the operator, and message ADAI41 is sent to the CP operator.

When a tape volume is successfully mounted, the message ADAI42 is displayed on the z/VM virtual console. This message also indicates whether the IDRC is in effect. If a cassette created with IDRC is used on a unit which does not support IDRC, an I/O error occurs.

If the volume parameters were supplied to DATADEF, the required volume serial number is included in both messages; if an output file is to be created and no volume serial number has been specified, the constant SCRTCH is displayed. This means that any unused standard label work (commonly called “scratch”) tape should be mounted. The two messages are repeated once each minute until the mount request has been satisfied. To reset the mount request, the z/VM operator must press the ENTER key twice. This causes open processing for the file to be terminated abnormally, and is equivalent to having

defined the file as ‘dummy’.

Tape Management with User Exit CMSUX1

You can control tape file management with a program that uses the optional user exit CMSUX1. If a program with the name CMSUX1 is loaded with the z/VM LOAD command before invoking a utility or nucleus that requires a tape file, the CMSUX1 program is called

- when the tape file is opened;
- whenever a tape volume must be mounted; and
- when the file is closed.

If a tape is rejected after a mount request because of a missing VOL1 record or incorrect volume/serial number, the CMSUX1 program is also reinvoked before a mount request is repeated.

When the CMSUX1 user exit receives control, general register 0 contains one of the following function codes:

```
1 for open
2 for tape mount
3 for close
```

General register 1 contains the address of a parameter list, which is passed to the CMSUX1 user exit program. The parameters are

- address of the data definition block (DDB) (non-modifiable);
- address of the six-byte tape volume serial number which will be requested (modifiable);
- address of a fullword containing:
  - an input/output flag byte (X‘00’ for input files, or X‘80’ for output files);
  - a reserved byte;
  - a two-byte hexadecimal tape unit address in the format X‘0*cuu*’ (ESA: X‘*ccuu*’). If the DATADEF statement specified TAP*x* in the UNIT parameter, this field contains the corresponding “18*x*” value according to standard z/VM tape unit conventions.
- address of a fullword containing the number of times the mount has been attempted (non-modifiable).

If the volume serial number has been modified, the new volume serial number is requested. If the file is an input file, however, this modification is not reflected in the DDB.

Upon return to the system, one of the following return codes is expected in general register 15:

```
0  proceed with the mount processing
4  proceed with the mount processing without issuing any operator messages
8  abort the mount processing
```

If return code 8 is encountered, the mount operation terminates as if an I/O error had occurred.

A sequential system file with the name of DUMP is available to facilitate dumping. This file may be assigned to printer, a z/VM file or DUMMY. If the file is DUMMY a CP dump will not be taken when an error occurs.

DATADEF File Assignments

All assignments for files which are accessed by an Adabas nucleus or utility must be done using DATADEF, which replaces the z/VM FILEDEF command in the Adabas environment. DATADEF accepts parameters as either a tokenized or extended parameter list; the extended parameter list takes precedence.

The file assignments established by the DATADEF statements can be listed using the DISPDD program. The file name of a specific DATADEF statement can be entered as a DISPDD parameter; if done, only the information for that file is displayed. If no parameter is specified, information for all assigned files is displayed.

The program RELDD can be used to clear active DATADEF entries. RELDD accepts a list of file names to be released or—if no list is specified—clears all active DATADEF entries.

The DATADEF statement creates a data definition block (DDB), which remains in system storage until it is

- overwritten by another DATADEF statement with the same name;
- cleared by RELDD; or
- cleared by a z/VM IPL.

The parameters for DATADEF consist of one positional parameter and one or more keyword parameters separated by commas. An equal sign (‘=’) must be used between a keyword and the parameter value. Depending on how DATADEF is invoked, spaces may be required surrounding equal signs, commas, and parentheses.

The DATADEF parameters are described in the following table:

Parameter Keyword	Required/ Optional	Maximum Length	Specifies . . .
positional	Required	8	the file name (DD) names as specified in *Adabas Operations* or *Adabas Utilities*.
BLKSIZE	See note 2	5	the length of the physical blocks in the file. If RECFM=FB, BLKSIZE must be an integral multiple of LRECL; if RECFM=V or RECFM=VB, BLKSIZE must be at least equal to LRECL + 4.

Parameter Keyword	Required/ Optional	Maximum Length	Specifies . . .
BUFNO	Optional	3 (1 - 255)	the number of buffers to be allocated for a sequential file on tape. Default: 5
COMPRESS	Optional	3 (YES - NO)	whether or not an output file on tape should make use of IDRC available on certain cassette units. If IDRC is not supported, this parameter is set to NO, the default.
CONCAT	Optional	3 (1 - 255)	a concatenation sequence number for the file. This results in the DDB being concatenated to another existing DDB with the same file name. The sequence numbers must be specified in ascending order with no numbers left out. The first file to be concatenated has the number 1. If specified for a file to be read backward, the sequence numbers are to be given in the normal sequential order. A DATADEF statement without CONCAT frees any existing root DDB and any DDBs concatenated to it.
DISP	Optional for output only	3	whether a sequential file is to be created (NEW), extended (MOD) or overwritten (OLD). If NEW is specified and the file exists a return code is issued. If OLD is specified, the file’s existence is not checked. Default: OLD
DSN	Required if not dummy	44	the data set or SFS directory name.
DUMMY	Optional	-	that the file does not exist. DUMMY is a keyword without a value. It may not be specified with any other parameter except file name.
EXTEND	Optional	-	that an existing DDB is to be extended. EXTEND is a keyword without a value. It may only be specified with the file name, VOL, and UNIT parameters.

Parameter Keyword	Required/ Optional	Maximum Length	Specifies . . .
FILESEQ	Tape only, optional	3	the sequence number of the file on a multi-file tape. The default value (1) is required if tapes are to be read backward.
FNAME	Required if DSN is a SFS	8	the file name of an SFS member.
FTYPE	Required if DSN is a SFS	8	the file type of an SFS member.
LRECL	See note 2	5	the length of the physical blocks in the file. If RECFM=FB, BLKSIZE must be an integral multiple of LRECL; if RECFM=V or RECFM=VB, BLKSIZE must be equal to or more than LRECL + 4.
MODE	See note 1	2	the z/VM filemode.
RECFM	See note 2	2	the format of the records in the file (F, FB, V, VB, U).
UNLOAD	Tape only, optional	3 (YES - NO)	whether or not the tape is rewound and unloaded. If NO is specified, the tape is rewound at close but is not unloaded. Default: YES.
UNIT	See notes 1 and 3	See note 1	a list of virtual addresses (*cuu*, or *ccuu* for ESA) of the unit or units containing the file, or one of the logical device abbreviations: TRM, PUN, RDR, PRT, or SFS. If a unit address list is given, it must be enclosed in parentheses and entries must be separated by commas.
VOL	See note 1	See note 1	a list of the serial numbers (each at most 6 characters) of the volumes containing the file; if multiple volumes are specified, they must be separated by commas and enclosed in parentheses.

Notes:

1. A MODE parameter is required for sequential z/VM DASD files. For DASD volumes containing database files, either a VOL, UNIT, or MODE parameter is required. If the database file spans multiple volumes, VOL or UNIT must be specified. Specifying MODE=* for an non-existent file results in a return code of 32.

If both VOL and UNIT are specified, the number of volumes and unit addresses must be equal and each volume in the VOL list must be mounted on the unit specified by the corresponding entry in the UNIT list (the first VOL entry must be mounted on the first UNIT entry, and so on).

For tape files, a UNIT must be specified and a real unit attached prior to DATADEF execution, unless the tape unit is dynamically allocated at open time; in this case, TAP*x* can be specified according to standard z/VM conventions, where TAP1 specifies virtual unit 181, TAP2 specifies unit 182, and so on. Only one tape unit address is allowed in the UNIT parameter.

The VOL parameter is required for input tape files, but is optional for output tape files. If a tape file spanning multiple volumes is to be read backwards, specify the volumes in the normal sequential order.

When creating a multi-volume tape file, Adabas z/VM maintains a list of the file volumes. To refer to that volume list in a later DATADEF, specify VOL=**filename* where *filename* is the name of the multi-volume file.

2. The parameters RECFM, LRECL, and BLKSIZE are required only for tape input files without a HDR2 label and for VSE sequential DASD files. If RECFM has been specified, the corresponding BLKSIZE and LRECL parameters are also required.

3. If UNIT=SFS, the DSN parameter is used as the SFS directory name and the FNAME and FYTPE parameters are used as the file name and file type, respectively. For example:

```
DATADEF DDCARD,DSN=SFSPOOL:USERID.DTR1,UNIT=SFS,FNAME=DB52,FYPTE=RUN1
or
DATADEF DDCARD,DSN=.DTR1,UNIT=SFS,FNAME=DB52,FYPTE=RUN1
or
DATADEF DDSAVE1,DSN=.DTR1,UNIT=SFS,FNAME=SSF,FYPTE=RUN001
```

DATADEF Error Codes

The following error codes may be returned by DATADEF:

Response Code	Description
16	No parameter list was supplied
20	Invalid keyword
24	No file name specified
28	Error in DSN or DUMMY specification: neither a DSN nor DUMMY was specified; or conflicting parameter specification for a dummy file
32	Error in VOL, UNIT or MODE parameter
36	Incorrect length for VOL,UNIT or MODE parameter
40	Insufficient virtual storage
44	Internal error issuing a CP command
48	Invalid *cuu* address (internal error)
52	Volume or unit not available or non-VTOC volume has been attached
56	Database file resides on volumes with mixed formats (VTOC, non-VTOC)
60	Database file resides on volumes of different device types
64	More than one unit specified for a tape file
68	Invalid file sequence number
72	Invalid RECFM parameter
76	Invalid BLKSIZE parameter
80	Invalid LRECL parameter
84	Invalid DISP parameter
88	Invalid concatenation count
92	Invalid DDB extension

These codes can be returned by invoking the CODES EXEC with the error number as a parameter.

Entering Operator Commands

Adabas operator commands can be entered directly from the virtual machine console of either the Adabas nucleus virtual machine, or from another virtual machine console authorized as a secondary console.

In addition, certain z/VM commands can also be entered as Adabas operator commands. This feature allows the database administrator to perform certain display or query functions while the Adabas nucleus is active. To enter a z/VM command, enter the command with the prefix CMS, followed by at least one blank before the command.

Caution:
If issued while Adabas is active, certain z/VM commands can have adverse effects on the nucleus, causing abnormal operation and even the erroneous data. Note also that issuing a z/VM command causes the nucleus to stop operation temporarily while the command is being processed, making Adabas unavailable during that time. The use of this z/VM command facility is solely the responsibility of the user; Software AG cannot accept responsibility for damage or loss that may occur when using this facility

Setting Defaults for ADARUN

Default values for the ADARUN parameters device type and database ID can be zapped into ADARUN at the offsets X'5D8' and X'8E8', respectively. For convenience, a file containing sample zap control statements to modify ADARUN has been supplied on the release tape. The file is named ADARUN ZAP. After making the necessary changes to this file, the user can enter the following commands to modify ADARUN:

```
ACC cuu fm
ZAP TXTLIB ADAVvvv (INPUT ADARUN
ACC cuu fm/A
```

where *cuu* is the virtual unit address of the Adabas library minidisk and *fm* is the z/VM filemode.

Installing New SM Levels

New SM levels may be installed by performing the following steps:

to install new SM levels:

1. Attach a tape unit to the database administrator virtual machine and mount a scratch tape.
2. Back up the existing library disk:

   ```
   TAPE DUMP * * fm (fm is the filemode of the Adabas library disk)
   TAPE RUN
   ```

3. Mount the system maintenance tape.
4. Enter the following commands:

   ```
   ACC cuu fm (cuu is virtual unit addr, fm is the filemode of the Adabas library disk)
   SMADA
   ```

5. If you have either created a user sublibrary, or if you have the ADARUN, DATADEF, RELDD or DISPDD modules on a commonly accessible minidisk, you must refresh the respective sublibrary and/or minidisk.

   Note:
   If you have installed a new maintenance level of Adabas, make sure that the APS*vrs* libraries were available when you refreshed the ADAECS module in the Adabas load library. If they were not, retry the refresh again with the APS*vrs* libraries. Otherwise problems might arise when you attempt to implement or use UES support for a database.

Applying Zaps in z/VM

In the z/VM environment, a zap should be applied using the ZAP command, to verify and replace data. The zap input control statements are in z/OS ZAP format. The ZAP command will apply the zap to a member of a z/VM TXTLIB.

The format of the ZAP command is:

```
ZAP TXTLIB libname (INPUT filename PRINT
```

where *libname* is the Adabas TXTLIB (ADAV*vrs*) and *filename* is a z/VM file with a filetype of ZAP that contains the input control statements.

The following are examples of ZAP control statements.

```
NAME membername csectname
VER disp data
REP disp data
END
LOG fixnum ZAPLOG text
* (COMMENT)
```

Note:
In the verify (VER...) and replace (REP...) statements, commas are acceptable data separators. However, commas with spaces or spaces alone are not acceptable data separators and can result in errors.

Refer to the IBM *z/VM Operator's Guide* for information about the ZAP service program.

IUCV Security Options

The *interuser communications vehicle* (IUCV) is a communication facility that allows one virtual machine to communicate with another virtual machine. Installations control the use of IUCV through the virtual machine directory entries.

In the Adabas environment, security on IUCV communications can be implemented by supplying different IUCV control statements. The following are examples of IUCV security options where:

IUCV ALLOW	is a general authorization indicating that any other virtual machine may establish a communications path with this virtual machine. No further authorization is required in the virtual machine initiating the communication.
IUCV ANY	is a general authorization indicating that a communications path can be established between this virtual machine and any other virtual machine.
IUCV *userid*	is the one- to eight-character user identification of the virtual machine with which this virtual machine is authorized to communicate.
idtmvmid	is the VMID of the ID table manager.
nucvmid	is the VMID of the Adabas nucleus.

No Security

```
ID Table Manager: IUCV ALLOW (PRIORITY)
IUCV ANY (PRIORITY)
Adabas Nucleus: IUCV ALLOW (PRIORITY)
DB Administrator: IUCV ALLOW (PRIORITY)
User: No IUCV CP directory entry required
```

Security on One Database

```
ID Table Manager: IUCV ALLOW
IUCV ANY
Adabas Nucleus: No IUCV CP directory entry required
DB Administrator: IUCV ALLOW
IUCV nucvmid
User: IUCV nucvmid
```

Total Security

```
ID Table Manager: IUCV ANY
IUCV ALLOW
Adabas Nucleus: IUCV idtmvmid
DB Administrator: IUCV idtmvmid
IUCV nucvmid
User: IUCV idtmvmid
IUCV nucvmid
```

Adabas 8 ADALNK/ADAUSER Installation 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*.

This section covers the following topics:

- 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)

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.

This section covers the following topics:

- Implementing the User Exits
- Registers and Processing
- Response Codes Returned

Implementing the User Exits

To implement the user exits, complete the following steps:

1. Edit LNKGBLS ASSEMBLE, and set the LX1NAME and LX2NAME parameters to your user exit file names. In addition, set parameters USERX1 and USERX2 to YES.
2. Assemble LNKGBLS.
3. Edit REFRESH EXEC. Add the user exits to the Adalnk INCLUDEs. For example:

```
'INCLUDE UEXITA'
'INCLUDE UEXITB'
```

4. Run the REFRESH EXEC to regenerate the ADALNK module.

Registers and Processing

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	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.

Response Codes Returned

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

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.

Adabas 7 ADALNK/ADAUSER Installation Considerations

Note:
We recommend that you install and use the Adabas 8 product and link routines.

This section covers the following topics:

- Adalink Considerations
- User Exit B (Pre-Command) and User Exit A (Post-Command)
- LNKUES for Data Conversion
- Creating the Adalink Module (ADALNK)
- ADAUSER Considerations

Adalink Considerations

Since link routines are dynamically loaded in most environments, it should only be necessary to replace the existing Adabas TXTLIB or TEXT files with the new Adabas TXTLIB, ADAV*vrs*. Programs that are either GENMODed or in a discontiguous shared segment (DCSS) must be regenerated.

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

One or two user exits may be linked with ADALNK:.

- UEXITB receives control before a command is passed to a target.

  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.

The user exits must be specified in the LOAD command to be active.

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).
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; for example, Adabas Review.

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 ADANUC user exits.

ADALNK can return the following non-zero response codes in ACBRSP:

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

The following two EQUates, described at the beginning of the source, can be modified before ADALNK is assembled. Other Adalinks allow this information to 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.

LNKUES for Data Conversion

The module LNKUES provides Universal Encoding Support (UES). This module must be 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.

Prior to Version 7, Entire Net-Work converted all data for mainframe Adabas. When Entire Net-Work Version 5.5 and above detects that it is connected to a target database that converts data, it passes the data through without converting it.

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 UEXITB.
- For replies, LNKUES receives control after UEXITA.

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.
- reassemble the translation tables and relink 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.

```
//LINK EXEC PGM=IEWL,REGION=0M
// PARM='XREF,REUS,LIST,LET,NCAL,SIZE=(1024K,256K)'
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DISP=SHR,DSN=USER.LOAD(ADALNK)
//ADALIB DD DISP=SHR,DSN=ADABAS.LOAD
//SYSLIN DD *
MODE AMODE(31) RMODE(ANY)
ENTRY ADABAS
INCLUDE ADALIB(ADALNK)
INCLUDE ADALIB(LNKUES)
INCLUDE ADALIB(ASC2EBC)
INCLUDE ADALIB(EBC2ASC)
NAME ADALNK(R)
/*
//
```

The (re)linked ADALNK must be made available to Entire Net-Work. If you are calling Adabas version 7 and you do not have the correct LNKUES/ADALNK module, Adabas produces unexpected results: response code 022, 253, etc.

Creating the Adalink Module (ADALNK)

Perform the following steps to create the ADALNK module:

to create the ADALNK module:

1. assemble ADALNK and UEXITA/UEXITB

   To assemble ADALNK, enter the following z/VM commands:

   ```
   ACC cuu1 filemode/A
   ACC cuu2 S
   GLOBAL MACLIB ADAV7vv MVSXA HCPGPI DMSGPI DMSOM
   ASMAHL ADALNK
   ```

   where *cuu*1 is the virtual unit address of the Adabas library minidisk, *cuu*2 is the virtual unit address of the system minidisk containing the MVSXA MACLIB, and *filemode* is the filemode for the Adabas library minidisk.

   To assemble UEXITA/UEXITB, enter the same commands as above:

   ```
   ACC cuu1 filemode/A
   ACC cuu2 S
   GLOBAL MACLIB ADAV7vv MVSXA HCPGPI DMSGPI DMSOM
   ASMAHL uexitname
   ```

where *cuu*1 is the virtual unit address of the Adabas library minidisk, *cuu*2 is the virtual unit address of the system minidisk containing the MVSXA MACLIB, *filemode* is the filemode for the Adabas library minidisk, and *uexitname* is the UEXITA/B filename (with a filetype of TEXT).

2. load ADALNK and the user exits, by entering the following z/VM commands:

```
LOAD ADALNK (RESET ADALNK RLD
INCLUDE uexitname (RESET ADALNK RLD
```

where *uexitname* is the filename (with filetype of TEXT) of UEXITA or UEXITB. To load both UEXITA and UEXITB, specify the INCLUDE command again for the second user exit.

3. generate a module, by entering the z/VM command:

```
GENMOD
```

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.

ADAUSER operates in the following way:

- ADAUSER is the only program that contains the entry point ADABAS. ADAUSER is dynamically loaded from the Adabas library, ADAV*vvv* TXTLIB, when a user program is loaded.
- On the first call to Adabas, ADAUSER invokes ADARUN, which is installed as a SYSTEM nucleus extension at LOGON time by the NUCXTNTS EXEC, and is release-independent. Subsequent Adabas calls bypass ADARUN.
- When invoked, ADARUN processes its control statements. If the ADARUN PROGRAM parameter has the (default) value USER, ADARUN loads ADALNK if the ADARUN MODE parameter specifies MULTI (the default), or ADANUC if the ADARUN MODE parameter specifies SINGLE. Therefore, ADAUSER is mode-independent.