software AG

Adabas

Installation for BS2000/OSD

Version 8.2.4

March 2012

Adabas

Table of Contents

1 Installation for BS2000/OSD

This document is intended for those who plan or perform Adabas installation for the BS2000/OSD operating system.

- *Supported Environments*
- *Installing Adabas for BS2000/OSD*
- *Installing Adabas With TP Monitors*
- *Device and File Considerations*
- *Installing the AOS Demo Version*
- *Installing the Recovery Aid (ADARAI)*
- *Adabas Dump Formatting Tool (ADAFDP)*
- *ADAUTM (Universal Transaction Monitor Support)*
- *Translation Tables*

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

2   Supported Environments

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

3 Installing Adabas for BS2000/OSD

This section describes the preparation for and installation of Adabas on systems running under the Fujitsu Technology Solutions BS2000 operating system.

Information for your specific installation is contained in the *BS2000 Product Installation Package* distributed with the System Installation Aid.

Installation Checklist

The following list provides an overview of the Adabas installation procedure on BS2000 systems.

Step	Description	Additional Information
1	Provide disk space for the Adabas libraries.	The libraries are restored from the installation tape into these libraries. Refer to the section *Adabas Library Disk Space Requirements*.
2	Allocate disk 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	Install the product license file (job I010, steps 0010 and 0011).	For more information, read *Step 1: Check, Prepare, and Install the Product License File (SMA job number I010)*
5	Allocate and format the Adabas database with the ADAFRM utility job (job I030, steps 0010 to 0018).	
6	Define the global database characteristics with the ADADEF utility job (job I030, steps 0020 and 0021).	
7	Start the Adabas nucleus and test the Adabas communications with the ADANUC job.	
8	Load the demonstration files ADA*vrs*.EMPL/MISC/VEHI/PERS/PERL with the ADALOD utility (performed in job I050 steps 0001 to 0003). **Note:** The Personnel demo file must be installed on a UES-enabled database because it includes wide-character format (W) fields.	
9	If appropriate, test Adabas address space communications by running ADAREP (job I050 step 9999).	
10	If appropriate, load the Adabas Online System (AOS) selectable unit into a Natural system file by running the ADAINPL job. Alternatively, install the AOS demo version delivered with Adabas (job I061, steps 0111 and 0112).	

Step	Description	Additional Information
11	Terminate the Adabas nucleus with an `ADAEND` operator command.	
12	Back up the database by running the ADASAV utility job.	

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 BS2000 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*.JOBS	The sample job 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*.MOD	The load library for 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.

Library Name	Description
ADA*vrs*.SRC	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*.SRC).
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*.L016	A Software AG internal library. The *vrs* in the library name represents the *version* of the internal library code, which is not necessarily the same as the version of Adabas.
APS*vrs*.PAMS	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*.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*.MOD	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*.MOD	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*.SRC	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

This section provides information related to activities required prior to Adabas installation.

- Defining a BS2000 Logon ID
- Libraries Required for UES Support
- Adabas Library Disk Space Requirements
- Disk Space Requirements for the Database
- Adabas Nucleus Address Space Requirements
- Disk Space Requirements for Internal Product Libraries
- Using DAB-Supported Volumes
- Migrating an Existing Database

■ Removing Version Numbers from Time Zone Library Elements

Defining a BS2000 Logon ID

Before you install Adabas in a BS2000 system, you must:

■ define a BS2000 logon ID for Adabas that permits loading from a release or installation tape;

■ enable all privileges for the logon ID to allow optional operations. This includes such capabilities as priority, T/P class, and maximum virtual memory size.

When allocating direct access files, private volumes are preferred to avoid the space fragmentation that can occur with public volumes.

Libraries Required for UES Support

The Software AG internal product libraries (APS - operating system layer) are required if you intend to enable a database for universal encoding service (UES) support. For UES support, the following library must be loaded and included in the BLSLIB concatenation: APS*vrs*.LIB (internal product application server), updated with the last patch level (APS*vrs*.L*nnn*).

> **Note:** Newer APS module libraries are updates of the original. We advise you to copy the original, merge the new library into it, and reference the merged library in your JCL. The new merged library must be provided twice, with the link-name APSLIB and with the link-name BLSLIB.

Also for UES support, the following library must be loaded and included with the link name DDECSOJ in the session execution JCL: ADA*vrs*.EC00, where *vrs* is the *version* of Adabas (for example, "812").

Adabas Library Disk Space Requirements

The Adabas files and libraries require the following disk space in PAM pages where *vrs* is the Adabas *version*:

File Name	Size	Description
ADA*vrs*.MOD	3468	Module library
ADA*vrs*.SRC	768	Adabas source, macros, and example jobs
WAL*vrs*.SRC	192	Adabas limited source, macros, and example jobs
WAL*vrs*.MOD	384	BS2000-dependent modules for Adabas limited
ADA*vrs*.EC*nn*	6432	UES code pages
ADA*vrs*.TZ00	672	Time zone elements

Disk Space Requirements for the Database

The Adabas database size is based on user requirements. For more information, refer to the *Adabas DBA Tasks* documentation. The following are suggested 2002 device cylinder and PAM page sizes for an initial Adabas database, allowing for limited loading of user files and the installation of Natural:

Database Component	Cylinders	PAM Pages
ASSOR1 (Associator)	50	7700
DATAR1 (Data Storage)	250	38000
WORKR1 (Work space)	50	7700
TEMPR1 (temporary work space)	25	3850
SORTR1 (sort work space)	25	3850

Adabas Nucleus Address Space Requirements

The typical Adabas nucleus requires at least 800-1024 KB of storage to operate. The size of the nucleus partition address space may have to be larger, depending on the ADARUN parameter settings. The address space parameters for the BS2000 user ID's JOIN entry must also be adequate. Parameter settings are determined by the user.

Disk Space Requirements for Internal Product Libraries

The minimum disk space requirements on the disk for the internal product libraries delivered with this version of Adabas are as follows:

Library	PAM Pages
ADA*vrs*.EC*nn*	6432
APS272.LIB	2304

Using DAB-Supported Volumes

The following restrictions apply to Adabas components when located on DAB-supported volumes:

Component	Recommendations/Restrictions
ASSO	not recommended for read caching; prohibited for write caching
DATA	not recommended for read caching; prohibited for write caching
WORK	no restrictions for read caching; prohibited for write caching
CLOG/PLOG/RLOG	no restrictions for read caching; prohibited for write caching
SORT/TEMP	no restrictions
Sequential input	no restrictions

Component	Recommendations/Restrictions
Sequential output	no restrictions

Migrating an Existing Database

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

Removing Version Numbers from Time Zone Library Elements

The time zone library for Adabas has been delivered containing elements with version numbers. Attempts to read these will result in the following error message:

```
ADANTZ .... Error Opending DD TZINFO(....)
```

To avoid this, remove the version numbers by issuing the following BS2000 commands:

```
MOD-FILE-ATTR ADAvrs.TZ00,AC=W
STA-LMS
O ADAvrs.TZ00,U
M-E-AT (,*,*),N=(V=*U)
END
MOD-FILE-ATTR ADAvrs.TZ00,AC=R
```

Installing the Adabas Release Tape

Adabas release tapes available to Software AG affiliates contain all Adabas product options. The affiliates use these release tapes to create custom installation tapes for customers according to contract agreements.

For specific information about your particular release tape, refer to the *Installation Notes* delivered with the installation tape.

- Installation Using SMA

■ Installation Not Using SMA

Installation Using SMA

If you are installing Adabas using the Software AG System Maintenance Aid (SMA), refer to the *System Maintenance Aid* documentation and to the information provided with the installation tape for specific installation instructions.

Installation Not Using SMA

> **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 files from tape to disk. You will then need to perform the individual installation procedure for each component to be installed.

- ■ Step 1: Copy the Library SRVvrs.LIB from Tape to Disk
- ■ Step 2: Copy the Procedure COPY.PROC from Tape to Disk
- ■ Step 3: Copy all Product Files from Tape to Disk

Step 1: Copy the Library SRVvrs.LIB from Tape to Disk

This step is not necessary if you have already copied the library SRV*vrs*.LIB from another Software AG installation tape. For further information, refer to the element #READ-ME in this library. The library SRV*vrs*.LIB is stored on the tape as a sequential file named SRV*vrs*.LIBS containing LMS commands. The current version *vrs* can be obtained from the *Report of Tape Creation*. To convert this sequential file into an LMS library, execute the following commands:

```
/IMPORT-FILE  SUPPORT=*TAPE(FILE-NAME=SRVvrs.LIBS,-
/   VOLUME=volser, DEV-TYPE=tape-device)
/ADD-FILE-LINK  LINK-NAME=EDTSAM, FILE-NAME=SRVvrs.LIBS,-
/   SUPPORT=*TAPE(FILE-SEQ=3), ACC-METH=*BY-CAT,-
/   BUF-LEN=*BY-CAT, REC-FORM=*BY-CAT, REC-SIZE=*BY-CAT
/START-EDT
@READ   '/'
@SYSTEM 'REMOVE-FILE-LINK  EDTSAM'
@SYSTEM 'EXPORT-FILE  FILE-NAME=SRVvrs.LIBS'
@WRITE  'SRVvrs.LIBS'
@HALT
/ASS-SYSDTA  SRVvrs.LIBS
/MOD-JOB-SW  ON=1
/START-PROG  $LMS
/MOD-JOB-SW  OFF=1
/ASS-SYSDTA  *PRIMARY
```

where:

tape-device is the device type of the tape, for example: TAPE-C4

volser is the VOLSER of the tape (see *Report of Tape Creation*)

Step 2: Copy the Procedure COPY.PROC from Tape to Disk

To copy the procedure `COPY.PROC` to disk, call the procedure `P.COPYTAPE` in the library `SRVvrs.LIB`:

```
/CALL-PROCEDURE  (SRVvrs.LIB,P.COPYTAPE), -
/   (VSNT=volser, DEVT=tape-device)
```

If you use a TAPE-C4 device, you may omit the parameter `DEVT`.

Step 3: Copy all Product Files from Tape to Disk

To copy all Software AG product files from tape to disk, enter the procedure `COPY.PROC`:

```
/ENTER-PROCEDURE  COPY.PROC, DEVT=tape-device
```

If you use a TAPE-C4 device, you may omit the parameter `DEVT`. The result of this procedure is written to the file `L.REPORT.SRV`.

Installing an Adabas Database

Once you have installed the Adabas installation tape, 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
- 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.

Database Installation Steps

- Step 1: Check, Prepare, and Install the Product License File (SMA job number I010)
- Step 2: Allocate and Format the Adabas Database (SMA Job Number I030)
- Step 3: Define the Global Database Characteristics (SMA Job Number I030)
- Step 4: Load the Demonstration Files (SMA Job Number I050)
- Step 5: Load Adabas Online System Add-On Product, If Appropriate (SMA Job Number I061)
- Step 6. Back Up the Database
- Step 7: Insert the ADARUN Defaults
- Step 8: Install the Required TP Link Routines for Adabas

Step 1: Check, Prepare, and Install the Product License File (SMA job number I010)

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 BS2000 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 BS2000 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 BS2000, as described in *Transferring a License File from PC to a BS2000 Host Using FTP*, in *Software AG Mainframe Product Licensing*.

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:

▶ **To install the product license file once it has been transferred:**

1   Run the LICMAKE license conversion job to load the license file into a library and assemble it.

2   Store the assembler output module ADALIC(R) in the job library and add this to your BLSLIB concatenation.

> **Note:** The Adabas 8 nucleus needs a license file to start up. Either the license file is assembled as an object by ADALIC(R) into a library in the DDLIB/BLSLIB chain or the license is read from the link name DDLIC. If neither of these is the case, the nucleus will end with a DMS0A4F error because it cannot find a link name on DDLIC.

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 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 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;
- 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 database ID and database device type(s);
- ADALOD parameters.

Step 5: Load Adabas Online System Add-On Product, If Appropriate (SMA Job Number I061)

Customize and run the job AOSINPL 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 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 6. 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 database ID and device type(s);
- ADASAV parameters.

Step 7: Insert the ADARUN Defaults

The member DEFAULTS in the Adabas job 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: 2300)
  - Database ID (default: 1)

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

Step 8: 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.

The Adabas BS2000 Communication Environment

The installation of a supervisor call (SVC) to provide a communication environment for Adabas is not required on BS2000 systems. Adabas uses the BS2000 executive services common memory pool and eventing for interprocess communication.

The router functions are implemented in the form of subroutines contained in the module ADARER. ADARER is loaded into common memory pool during nucleus initialization, and is shared by all user tasks that issue Adabas calls.

For those systems running an earlier Adabas/WAL library but that wish to run with the Version 8.2 environment, the Version 8.2 system must be started first in order to create the version 8.2 router (ADARER) required for this version's nucleus.

Considerations of Device Types

The new default device type is 2300. All devices have standard block sizes that are a multiple of 4 and rounded to a 2K boundary. Thus they can be used on NK4 disks and D3475-8F devices.

If you do not wish to migrate the default device used by prior releases of Adabas (device 2000) to the new default (device 2300), then the JCL will need to include `DEVICE=2000` in the ADARUN cards.

Devices and Spanned Records

Spanned record files are those loaded with the ADALOD SPAN option.

Writing a record over a block boundary in such a file requires an extra 50 to 100 bytes in the Work file block. In such cases, the Work file needs to have a block size of 50 to 100 bytes larger than the Data block size.

The default device 2300 has Data and Work block sizes of 8192 bytes. Therefore, there are not enough spare bytes in default device 2301 to span records over blocks. Device 2301, however, has Data and Work block sizes of 16384 bytes, so this device can be used.

If the default device must process spanned records, the Work file block should be formatted, for example, as a 2301 device at block size 16384, and the ADADEF database define should set WORKDEV=2301. A zap file will be made available at a later time to address this workaround.

Applying Zaps

Every effort has been made to make Adabas operating system independent. However, Adabas development is done in an IBM environment and corrections and changes are prepared in this environment.

As a result, corrections are applied to Adabas modules with IMASPZAP, which determines the syntax of any zaps. When corrections must be applied to a module library at BS2000 locations, the zaps must be revised to the LMS format. The following example shows how this can be done:

▶ **To revise a zap to LMS format:**

1   Assuming a zap in the following LMS format:

```
/EXEC LMS
LIB ADAvrs.MOD,BOTH
UPDR ADAEXAMP (a)
*COR ADAEXAMP,12BC,X'47B0A123'=X'4720BAEE' (b,c)
*END
END
```

where:

a specifies corrections to module ADAEXAMP.

b verifies the contents at location X'12BC'. The correction is applied only if the content of this location is X'47B0A123'.

c specifies a replacement value X'4720BAEE' for the location X'12BC'.

2   To verify the original data at location X'12BC' on the BS2000 system, perform the following:

```
/LOAD (ADAEXAMP,ADAvrs.MOD) Load module
/DISPLAY L'12BC'.(L=4) Display original content
%D C=ADAEXAMP.H'12BC'%XL4 (AID)
```

If the data displayed is different from the value given on the original VER statement, do not continue with step 3.

3   Run the LMS procedure.

4   To ensure that the last step was performed properly, perform step 2 again. This procedure should also be used for ADARUN and ADALNK zaps, described later in this section.

Job Variable (JV) Handling

The control job variable (JV) is filled automatically if the statement

```
/DEL-JV name
/SET-JOB-STEP
/CRE-JV name
/SET-JV-LINK *ADA,name
```

—is contained in the nucleus startup job control statements. The JV layout is as follows:

Positions	Length	Representation	Contents
1 - 128	128	undefined	unused
129 - 136	8	alphanumeric	program name
137 - 140	4	numeric	user abend code
141 - 145	5	numeric	error number (parameter or utility)
146 - 150	5	numeric	Adabas response code
151 - 157	7	numeric	CPU time, program-related (sec.)
158 - 158	1	alphanumeric	blank
159 - 165	7	numeric	elapsed time, program-related (sec.)
166 - 178	13	alphanumeric	blank, reserved

Job Switches

Adabas uses job switch 10. The job switch is set by ADARUN and reset if the nucleus or utility session terminates normally. It can be used for session control to indicate whether or not a termination is normal. When ADARUN is called with `PROGRAM=USER`, no switches are set or reset.

Linking User Exits B and A to ADALNK

One or two user exits may be linked to or loaded with ADALNK:

- UEXITB (pre-command) receives control before a command is passed to a target by the router 04 call.

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

- UEXITA (post-command) receives control after a command has been completely processed by a target, the router, or by ADALNK itself.

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

Register	Content
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
14	Return address
15	Entry point address: UEXITA or UEXITB

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, 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 (ADARSP216) 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 Optimizer Exit OPTEXIT

An exit in ADALNK for the Natural Optimizer is provided in BS2000 environments. The exit is called before the UEXITA (user exit A). The source module OPTEXIT(S) is supplied in the ADA*vrs*.SRC library and is already bound to the supplied ADALNK. If it is to be changed, it will have to be replaced.

Enabling Universal Encoding Support (UES) for Your Adabas Nucleus

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

For Adabas Version 8.2, UES is enabled by default for the link routine ADALNK.

> **Note:** The use of UES-enabled link routines is transparent to applications, including applications that do not require UES translation support: it is not necessary to disable UES support.

- Load Module and Translation Tables
- Job Steps
- Calling LNKUES

Load Module and Translation Tables

No source module for ADALNK is delivered with this version of Adabas. The module ADALNK already has the UES component, LNKUES, and the translation tables ASC2EBC and EBC2ASC bound to it.

The ADALNK module consists of ADALNK, ADAL2P and LNKUES. You can bind modified ASC2EBC or EBC2ASC translation tables to ADALNK-BASE using the job ALNKUES.

The two standard translation tables are:

- ASC2EBC: ASCII to EBCDIC translation; and

- EBC2ASC: EBCDIC to ASCII translation.

It should only be necessary to modify these translation tables in the rare case when a 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.

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

Job Steps

Job library member ALNKUES is an example job of how to link ADALNK with the UES components.

Calling LNKUES

LNKUES is called only on ADALNK requests (X'08'), (X'0C') with reply (X'10') or (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.

- Required Environment
- Connection Possibilities
- JCL Required for UES Support

Required Environment

The Adabas database must be UES-enabled. See *Adabas DBA Tasks* and the ADACMP and ADADEF utilities in *Adabas Utilities* for more information.

Connection Possibilities

UES-enabled databases are connected to machines with different architectures through Software AG's internal product software (APS) or through Entire Net-Work (WCP).

Adabas SQL Gateway (ACE) clients may not be strictly EBCDIC in an environment where databases are connected through Software AG's internal product software (APS).

JCL Required for UES Support

The nucleus job statements for a UES nucleus require the following items:

■ BLSLIB access to the APS module libraries is required, so these extras statements are required:

```
/SET-FILE-LINK APSLIB,$SAG.APSvrs.LIB
/SET-FILE-LINK BLSLIBnn,$SAG.APSvrs.LIB
```

> **Note:** Newer APS module libraries are updates of the original. We advise you to copy the original, merge the new library into it, and reference the merged library in the JCL.

■ The job needs to contain the procedure call below to access the DDECSOJ object library:

```
/CALL-PROCEDURE ($SAG.APSvrs.LIB,LMSLINKLIB),(LNK-NAME=BLSLIBn1)
```

---where BLSLIBn1 is last in the BLSLIB sequence

■ The DDECSOJ SET-FILE-LINK statement should point to the ECS encoding objects $SAG.ADAvrs.ECnn library.

■ No special Software AG internal product software (APS) parameters need to be set.

The setting of extra options on the START-PROGRAM statement can be done as follows:

```
/START-PROGRAM ($SAG.ADAvrs.MOD,ADARUN),-
/ RUN-MODE=*ADVANCED(ALT-LIB=YES,LOAD-INF=*REF,UNRES=*DELAY,-
/ MESSAGE=*ERROR)
```

■ An example of Adabas session job control for UES (BS2000) is supplied in the $SAG.ADAvrs.SRC(ADANUCU,J) library element.

> **Note:** The Software AG internal product software (APS) batch system will create a subtask job with the same user attributes as the UES nucleus job. This job will be stopped automatically when the UES nucleus is stopped.

Linking LNKUES to ADALNK for Data Conversion

Adabas Version 7 was delivered with the module LNKUES for Universal Encoding Support (UES). LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller. In Adabas version 7, LNKUES had to be linked to ADALNK; in version 8, LNKUES is fix bound to ADALNK.

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 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.

```
/ASS-SYSDTA *SYSCMD
/STA-PROG $TSOLNK
MODULE ADALNK,LIB=USER.MOD,ELEM=ADALNK
NCAL
LINK SYMBOLS *KEEP
INCLUDE ADALNK,ADABAS.MOD
INCLUDE ASC2EBC,ADABAS.MOD
INCLUDE EBC2ASC,ADABAS.MOD
```

```
BIND
/ASS-SYSDTA *PRIM
```

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 (ADARSP022), 253 (ADARSP253), etc.

Formatting New Adabas Libraries

By formatting a database container, the size of it is physically defined. The following rules apply:

1. The highest PAM Page of the database container must be less than the HIGH-US-PA allocation in the `SHOW-FILE-ATTRIBUTES` *container*,`ALL` command output. Otherwise, accessing those pages higher than that physical limit will result in an I/O error DMS0922. For more information, read *Device and File Considerations*, elsewhere in this guide.

2. The Associator (ASSO) must have a top RABN that is greater than 30. Data Storage (DATA) must have a top RABN that is greater than 2.

3. The SORT and TEMP containers need not be formatted *if* they are being used with the access mode OUTIN.

Loading Elements into the Time Zone Library (TZ00)

The time zone library is delivered with Adabas as ADA*vrs*.TZ00 and contains many elements of type X. If you need to load new elements into this library, modify and run job TZUPLOAD provided in ADA*vrs*.SRCE. This job can be adapted to load the new time zone element into the library in the correct format.

The loaded time zone element should contain the default version number (*U). If it does not, the following error occurs:

```
ADANTZ .... Error Opening DD TZINFO(....)
```

To resolve this problem, read *Removing Version Numbers from Time Zone Library Elements*, elsewhere in this guide.

Starting Adabas

Adabas User Exit 2 for BS2000 permits two methods for specifying the /ENTER-JOB job:

- using a job variable containing the complete /ENTER-JOB job command;

- defaulting to an ENTER-JOB RES.E.*x*LCO startup command.

For more information, see the description of *User Exit 2* in the user exit and hyperdescriptor exit documentation.

Interpreting BS2000 Error Messages

When running Adabas, some BS2000 messages may indicate that a parameter is missing or is incorrect. The following are the message IDs that can occur, and their explanations in an Adabas environment:

Message	Description
D922	SAM file allocation (for example, for DDOUT1) primary and secondary allocation must be a multiple of the both standard block size (16) and of the allocation unit (3). The smallest multiple allowed is 48. The standard block size may have been changed using the ADARUN parameter `QBLKSIZE`. If so, the standard block size is determined by dividing the sum of the `QBLKSIZE` value and 2047, by 2048.
DD99	Parallel access to a PAM file (for example, DATA) was attempted, but `SHARUPD=YES` was omitted
DDB1	Refer to the D922 description above
DDC2	`SHARUPD=YES` was specified for a SAM file (for example, DDSIBA)

4 Installing Adabas With TP Monitors

This section provides information needed to install Adabas with teleprocessing (TP) monitors TIAM and UTM.

The Adabas API for BS2000

The Adabas Version application programming interface (API) comprises the modules:

- ADAUSER (delivered in source)

- ADAUSER2 (delivered in source)

- ADALNK (*not* delivered in source, but the ADALNK module is comprised of the components ADALNK.ADAL2P and LNKUES and can be used for binding modified ADALNK components)

- SSFB2C (delivered in source, plus macro B2CONFIG to customize the interface)

ADALNK contains the combined functionality of all the ADALN*x* modules of previous Adabas versions. Software AG recommends that you link ADAUSER to the application program and use the entry point ADABAS on the API call.

For compatibility with existing applications, ADALNK contains the following entry points:

Entry Point	TP Monitor
ADALNK	Batch / TIAM
ADALNR	Batch / TIAM (reentrant)
ADALNN	UTM running Natural
ADALNU	UTM with Assembler or a 3GL language

ADALNK is reentrant. ADAUSER, which is supplied as source to allow modification, can be made reentrant if the user application provides a 4-byte anchor area.

Because ADALNK is reentrant, parameter items previously moved into ADALNK are now found in SSFB2C.

The symbol "ADABAS" is no longer in ADALNK, but in ADAUSER.

The symbol "ADALNR" is offset X'C' (decimal 12) bytes into the start of ADALNK. If fields are to be accessed in this version of ADALNK, their offsets have changed accordingly.

- ESD Symbols Used in the Adabas API
- ADAUSER Program
- ADAUSER2
- Fixed Linking of ADALNK or ADAUSER
- Routing and Adabas Review Parameters
- Presetting Parameters in SSF2BC

- ADALNK
- ADALNK Access to Multiple ID Tables
- ADALNK Parameter Service
- Assembling ADALNK

ESD Symbols Used in the Adabas API

The following symbols have a special meaning in the Adabas API:

Symbol	Meaning
ADAUSRID	identification area used by Natural; identifies the caller as Natural
AUTUSRID	identification area used by ADAUTM
CMSTART	identifies the batch Natural driver
KDCKB	area in UTM holding the user/task ID
KDCUTMD	identifier that the carrier is UTM
KDCUTMID	alternative identifier that the carrier is UTM

Software AG advises you not to define or write-access any of these symbols in applications that also use the Adabas API.

ADAUSER Program

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.

Software AG recommends that you link ADAUSER to your applications because it:

- can be made reentrant;
- is small; and
- has no version-specific functions. In future versions, application link jobs will not require changes. For example, the application and its linkage will not be affected if TCP/IP is used to access the database.

ADAUSER propagates symbols to the loaded API interface, which can be used to pass user data or determine the carrier system the application is using. ADALNK does not do this. If ADALNK is to be linked, the whole API must be linked to the application.

- ADAUSER Loading ADALNK

- ■ ADAUSER Loading ADARUN

ADAUSER Loading ADALNK

The following file statement assigns a SAM/V file for the ADALNK parameters:

```
/SET-FILE-LINK DDLNKPAR,samv-file
```

If the file link DDCARD is specified and the application is not running on UTM, ADAUSER loads ADARUN. This is the traditional API.

If the file link DDCARD is not present, ADAUSER loads ADALNK. In this case, the routing information (`idt_name,dbid`) and the Adabas Review buffer size can be defined in the file link DDLNKPAR.

ADAUSER Loading ADARUN

If the file link DDLNKPAR is not specified, ADAUSER loads ADARUN for using prefetch and multifetch. The routing and Adabas Review buffer parameters are then to be delivered in the file link DDCARD or *SYSDTA.

The following file statement assigns the proper Adabas module library `modlib`:

```
/SET-FILE-LINK DDLIB,modlib
```

ADAUSER can load modules from several libraries. If the ADARUN-defined library is found in the catalog, it becomes the primary library.

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

ADARUN processes its control statements:

- ■ If ADARUN `PROGRAM=USER` (the default) and ADARUN `MODE=MULTI` (the default), ADARUN also loads ADALNK.
- ■ If ADARUN `PROGRAM=USER` (the default) and ADARUN `MODE=SINGLE`, ADARUN also loads ADANUC.

This makes ADAUSER mode-independent.

ADAUSER2

ADAUSER2 is a program that allows Natural to load a COBOL module where Natural and COBOL have separate Adabas session IDs. ADAUSER2 is delivered in the ADA*vrs*.SRC library and must be assembled before it can be used.

Fixed Linking of ADALNK or ADAUSER

To resolve external symbols in the application module, link the following modules to it:

- for ADALN*x* symbols, ADALNK-BASE, ASC2EBC, EBC2ASC, and SSFB2C
- for the ADABAS symbol, ADAUSER (note that the delivered ADAUSER is not reentrant)

If you are linking ADALNK-BASE, ASC2EBC, EBC2ASC, and SSFB2C to the application module and you need to resolve the ADABAS symbol as well, you can alternatively do one of the following:

- Insert the TSOSLNK command RENAME in the application link job to rename ADALNK before you include it:

```
RENAME ADALNK,ADABAS
INCLUDE ADALNK,$SAG.ADABAS.MOD
```

- To avoid changing the link job, insert the following lines behind the label ADALNU in the supplied ADALNK source in the ADA*vrs*.SRC library:

```
ENTRY ADABAS
```

Routing and Adabas Review Parameters

Routing parameters are used to locate the application's target database:

```
IDTNAME=
```

If ADALNK has been linked or loaded, these parameters are found under the link name DDLNKPAR. The ADAL2P component of ADALNK (ADALNK.ADAL2P) reads the data from DDLNKPAR and holds it in SSFB2C.

If ADAUSER has been linked or loaded and

- the file link DDCARD is present, ADARUN is loaded and these parameters are read from DDCARD. This is the traditional Adabas API.
- the file link DDCARD is not present, ADALNK is loaded and these parameters are found in DDLNKPAR.

ADARUN must be loaded in order to load and install the Adabas prefetch components:

```
ADARUN PROG=USER,PREFETCH=YES
```

Presetting Parameters in SSF2BC

Routing, Adabas Review, and some configuration information can be preset in SSFB2C using the delivered source module in ADA*vrs*.SRC and the macro B2CONFIG from ADA*vrs*.MAC:

SSFB2C Parameter	Description
IDTNAME=ADA*xxxxx*	ID table where the database runs

For example, the following presets routing to database 99 on the IDTNAME ADA00009:

```
SSFB2C CSECT
B2CONFIG DBID=99,IDTNAME=ADA00009
END
```

ADALNK

When loaded, ADALNK attempts to read and process control statements from its parameter file. Entries in the parameter file appear in the following format:

```
ADALNK IDTNAME=ADA12345
```

The ADALNK control statements are delivered with the default values specified in the following discussion. Changing default settings by using zaps or by re-assembling ADARUN and ADALNK should be the *rare exception*.

> **Note:** In a future version of Adabas, it will not be possible to change ADARUN and ADALNK parameters with zaps.

ADALNK Parameters

These parameters are those set in the link name DDLNKPAR.

> **Note:** Do not specify ADALNK statements in ADARUN nuclei, Entire Net-Work, ADAUSER, or utility contexts. If you do, a warning message (ADAK06) is reported, the ADALNK statements are ignored, and processing continues.

Parameter Syntax	Description
`DBALIAS={`*`nnnnn`*` | 0 |` ↵ `(`*`nnnn1`*`[,`*`nnnn2`*`]...)}`	The alias database ID, which will be converted into the corresponding database ID in the DBID= list and routed to the ID table in the DBTIDT list. The maximum number of elements in a DBALIAS list is 16.
`DBID= {`*`nnnnn`*` | `<u>`1`</u>`  |` ↵ `(`*`nnnn1`*`[,`*`nnnn2`*`]...)}`	Sets the default database ID. The minimum value is 1; maximum value is 65535. The default is 1. If multiple entries are specified, the first is the default database ID in the ID table defined in the IDTNAME ADARUN parameter; the other elements are routed to the ID table defined as parallel elements in the DBTIDT list. The maximum number of elements in a DBID list is 16.
`DBTGRP={ YES | NO |` ↵ `(`*`y`*`|`*`n`*`[,`*`y`*`|`*`n`*`]...)}`	The scope of the corresponding ID table name in the DBTIDT list. YES means GROUP scope; NO (or if undeclared) indicates that the scope is GLOBAL. The maximum number of elements in a DBTGRP list is 16.
`DBTIDT= (`*`ADAcccc`*` | 0 |` ↵ `(`*`ADAcccc1`*`[,`*`ADAcccc2`*`]...)}`	One or a list of additional ID table names that can be accessed from this ADALNK. The parallel element in the DBTGRP parameter defines the scope of the ID table. The maximum number of elements in a DBTIDT list is 16.
`DISACLOS={ YES | `<u>`NO`</u>` }`	Determines whether or not ADALNK should close all communication after executing the CL command to the last connected database so that the user task can issue BS2000 WRCPT macros (not available for ADALNN, ADALNR, or ADALNU Adalinks). The default is NO.
`GROUPS={ YES | `<u>`NO`</u>` |` ↵ `(`*`y`*`|`*`n`*`[,`*`y`*`|`*`n`*`]...)}`	Determines whether or not the communication items are valid for all logon user IDs with no restriction (NO, the default) or whether they are valid only for the logon user ID of the present task (YES). Se also the ADARUN GROUPS parameter in *Adabas Operations*. For multiple entries, the GROUPS element corresponds to the IDTNAME element in the IDTNAME= list. The maximum number of elements in a GROUPS list is 16.
`IDTNAME={ `<u>`ADABAS5B`</u>` ↵ | `*`ADAcccc`*` |` ↵ `(`*`ADAcccc1`*`[,`*`ADAcccc2`*`]...)}`	Specifies the name(s) of the ID table(s) to be accessed. This name must be the same as the nucleus' ADARUN `IDTNAME` parameter, if specified, which must be 8 characters beginning with "ADA". The default is ADABAS5B. The maximum number of elements in an IDTNAME list is 16.
`LRVINFO=`*`nnnnn`*	Sets the size of the Review buffer. The default is the setting in the B2CONFIG macro in SSFB2C (default 0).
`LU=`*`nnnnnn`*	Sets the size of the user buffer. The default is 0.
`REVHUB=`*`nnnnnn`*	Sets the Adabas Review hub ID to be accessed on the local machine. The default is 0 (none).
`X48={ YES | `<u>`NO`</u>` }`	Determines whether or not X48 call processing is available to an application so that it can build its own 28-byte communication ID to identify internal Adabas resources used by a logical transaction.

Notes:

1. Database IDs are in the range from 1 to 65535.

2. ID table names are up to 8 alphanumeric characters and must begin with "ADA".

3. The maximum number of elements in a list is 16.

ADALNK Access to Multiple ID Tables

You can route data to specific database IDs to other ID tables, and optionally, to change the database ID to another in the target ID table.

This can be accomplished in either of two ways:

1. In the file attached to the link name DDLNKPAR, set the DBTIDT parameter to declare additional ID tables and expand the DBID parameter settings to specify corresponding database IDs. For example:

```
DBTIDT=(ADAIDT02,ADAIDT03,...)
DBID=(5,360,...)
```

   In this example, calls to database 5 are in ID table ADAIDT02 and calls to database 360 are in ID table ADAIDT03.

   In addition, you can use the DBALIAS parameter to change one database to another before routing. This provides a means of accessing database IDs with the same number on different ID tables (perhaps for database replication). For example:

```
DBTIDT=(ADAIDT02,ADAIDT03,...)
DBID=(5,360,...)
DBALIAS=(,500,...)
```

   This example changes database ID 360 to 500 before routing.

2. Use the IDTABEL macro to define the alternative ID tables and the MDBIDT macro to link the database ID (and the ALIAS operand-specified database ID) to the ID table declared in IDTABEL or ENVNAME (from the BDCONFIG macro). Routing to a database ID declared in MDBIDT will go to the ID table declared in the same macro; the database ID will be changed to the ALIAS operand-specified ID if it is declared before routing. Then reassemble the SSFB2C parameter repository module. For more information, read *ADALNK Parameter Services*, elsewhere in this section.

ADALNK Parameter Service

The ADALNK parameter service coordinates ADARUN, ADALNK, and SSF parameters. It comprises three modules:

1. ADAL2P component of ADALNK (ADALNK.ADAL2P) contains the parameter service routines

2. the ADALNK module contains the Adalink parameters

3. SSFB2C contains operating system specific parameters. Most of the operating system related defaults (such as the names, scopes, and sizes of memory pools, serialization and event items) are concentrated in this module.

   The parameters in the delivered SSF2BC module contain the same defaults as associated ADARUN and ADALNK parameters and can be modified with the help of the ADARUN or ADALNK parameters. In some cases, it is desirable to hide these parameters.

   Starting with Adabas 8.2, ADALNK can access more than one ID table. Three macros are provided in SSFB2C to support this:

   - **B2CONFIG**, which specifies the ADALNK configuration parameters

   - **IDTABEL**, which specifies other IDTNAMEs to be accessed

   - **MDBIDT**, a cross-reference table of DBIDs with corresponding IDT names.

     - Linking ADALNK
     - B2CONFIG Macro
     - IDTABEL Macro
     - MDBIDT Macro
     - Parameter Priority
     - ADALNK Protocol File
     - Message Protocol
     - Example

Linking ADALNK

The ADALNK parameter service considers the SSFB2C configuration module:

- if ADALNK is loaded by ADAUSER, SSFB2C is AUTOLINKED by V-Constant. In this case, no changes are required.

- if ADALNK needs to be bound to a user program (this is not recommended), include the following statements in the TSOSLNK JCL that links ADALNK:

```
...
INCLUDE ADALNK
INCLUDE ASC2EBC
INCLUDE EBC2ASC
INCLUDE SSFB2C
or BINDER JCL :
...
INCLUDE-MODULES ELEMENT=(ADALNK,ASC2EBC,EBC2ASC,SSFB2C),-
...
```

B2CONFIG Macro

If other *defaults* are required, the SSF configuration module can be re-assembled. The macro B2CONFIG is delivered for this purpose.

The following operands are available:

Operand Syntax	Description
`ALNKPRT={ NO	YES }`	Permits or inhibits the creation of the ADALNK parameter services' protocol data set.
`DBID= {nnnnn	1 }`	Sets the default for the ADARUN and ADALNK parameter `DBID=`n. The maximum value is 65535; the default is 1.
`DISACLOS={ YES	NO }`	Sets the default of the ADALNK parameter `DISACLOS`.
`ENVNAME=cccccccc`	Sets the default for the ADARUN and ADALNK parameter `IDTNAME`. The operand should contain up to eight alphanumeric characters starting with the characters "ADA".
`ENVSCOPE= {GROUP	GLOBAL }`	Sets the default for the ADARUN and ADALNK parameter `GROUPS`. Specify: ■ `ENVSCOPE=GROUP` for ADALNK `GROUPS=YES` ■ `ENVSCOPE=GLOBAL` for ADALNK `GROUPS=NO`
`IDTSUP={Y	N}`	Indicates whether or not the output of the IDTNAME in the protocol should be suppressed.
`LRVINFO={nnnnn	0}`	Specifies the size of the Adabas Review ADALNK buffer in bytes.
`LUINFO={nnnnn	0}`	Specifies the size of the user buffer transferred with ADALNK.
`NUMGES={nnn	6000 }`	Determines the maximum number of global event control blocks allowed.
`NUMLID={nnn	27}`	Specifies the maximum number of entries in the local ID table. This means, the maximum number of database IDs that this application can concurrently access.

IDTABEL Macro

The IDTABEL macro declares other possible IDTs that can be routed to. The following operands are available:

Operand Syntax	Description
IDT=*iiiiiiii*	Specifies an IDTNAME (up to 8 bytes long) in addition to the ENVNAME identified in the B2CONFIG macro.
SCOPE={GLOBAL | GROUP}	Specifies the scope of the IDTNAME identified by the IDT operand. The default is GLOBAL.

MDBIDT Macro

The MDBIDT macro specifies cross-links between database IDs and ID tables. The following operands are available:

Operand Syntax	Description
IDT=*iiiiiiii*	Specifies an IDTNAME (up to 8 bytes long) which should have been declared in the B2CONFIG ENVNAME operand or in one of the IDTABEL macros.
DBID={*nnnnn* | ↵ (*nnnn1*[,*nnnn2*]...)}	Specifies the database IDs to route to the IDTNAME given in the IDT operand.
ALIAS=*nnnnn*	Specifies an alias database ID. Calls to a database in the DBID operand will be routed to the database identified by the ALIAS operand on the ID table referenced in the IDT operand.

📄 **Notes:**

1. Database IDs must be in the range 1 to 65535.

2. ID table names can be up to 8 alphanumeric characters long and must begin with the characters "ADA".

3. The maximum number of elements in the MBDIDT lis is 200; for other lists it is 16.

Parameter Priority

Parameter values changed by zaps take priority over the corresponding SSFB2C.

Parameter values set by ADARUN or ADALNK statements take priority over both values changed by zaps and values from SSFB2C.

Parameter values changed by zaps are not reported on the protocol file.

ADALNK Protocol File

The ADALNK parameter service allows you to protocol its statements

- into SYSLST (ASS-SYSLST library);

- into a file determined by "/SET-FILE-LINK DDPLNPRT, *file*" (if this link name is not included in the tasks file table, the protocol is written to SYSLST); or

- nowhere. In this case, the module SSFB2C must be assembled by setting the operand `ALNKPRT=NO`.

Message Protocol

```
ADAK04 CONFIGURATION MODULE FOUND :
ADAK04 NAME : USERCONF; ASS-DATE 960229; VERSION 0130    (see note 1)
ADAK04 ENVNAME =ADATEST1    (see note 2)
ADAK04 ENVSCOPE =GLOBAL    (see note 2)
ADAK04 DBID =145    (see note 2)
ADAK04 LRVINFO =0  (see note 2)
ADAK04 DISACLOS =NO    (see note 2)
ADAK04 THE FOLLOWING ADALNx PARAMETERS ARE IN USE FOR THIS RUN
ADALNK DBID=196   (see note 3)
```

> **Notes:**

1. Information to identify a user-defined configuration module.

2. B2CONFIG macro parameters.

3. ADALNK parameter read from the ADALNK parameter service.

4. Values changed by zaps are not reported.

Example

In the following example, DBIDs 63, 66, and 68 are routed to IDT ADABASSN, while DBID 70 is routed to DBID 69 on IDT ADASDE01. All of the DBIDs will be routed to ADASDE02. IDT ADASDE01 has GROUP (not GLOBAL) scope.

```
*************************************************
*   SSFB2C - CSECT storage area for ADABAS
*           non-reenterable
*************************************************
        B2CONFIG MF=C,ENVNAME=ADASDE02
*************************************************
        IDTABEL IDT=ADABASSN
        IDTABEL IDT=ADASDE01,SCOPE=GROUP
        MDBIDT  IDT=ADABAS5N,DBID=(63,66,68)
        MDBIDT  IDT=ADASDE01,DBID=69,ALIAS=70
        END                                   ↵
```

Assembling ADALNK

No ADALNK source is delivered with Adabas 8. To customize this interface, use UEXITB and UEXITA, as described in *Other Exits Supported by Adabas*, in the *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual*.

Installing Adabas with Batch / TIAM

The Adalink used for batch and the TIAM TP monitor environment under BS2000 is ADALNK.

The ADALNK entry point is used where a single user issues only one Adabas call at a time and waits synchronously on the call to return. The last eight bytes (UID) of the communication ID are set to either

- B*xxxx* (batch); or

- D*xxxx* (TIAM)

where *xxxx* is the BS2000 task sequence number.

This Adalink is loaded by ADARUN to control multiuser (ADARUN MODE=MULTI) or utility sessions. The CSECT name is ADALNK.

- ADALNK for SAPR Application Packages

- Dynamically Loading Symbols in Batch / TIAM

ADALNK for SAPR Application Packages

The ADALNR entry point is located at offset '*x*C' in ADALNK, which is reentrant.

Calls to ADALNR require an eighth Adabas call parameter containing the address of this initialized area, which must be below the 16-megabyte line.

The seventh parameter, which is reserved for Natural applications, must always be defined. This parameter must always be used for each Adabas call.

The addressed area itself, which cannot be changed by the user, must equal the total of the two halfword counts found at ADALNK locations X'B4' and X'B6'.

Dynamically Loading Symbols in Batch / TIAM

The binder/loader/starter (BLS) may be used to dynamically load the ADALN*x* (where *x* is K, N, Q, R, or U) or ADABAS symbol with the job statements

```
/SET-FILE-LINK DDLIB,<adabas_library>
.
/START-PROGRAM (MY.LIB,MYAPPL),RUN-MODE=ADVANCED(ALT-LIB=YES)
.
```

where *adabas_library* is the name of the delivered Adabas module library.

If the program MYAPPL has the unresolved external symbol:

- ADALN*x*, then ADALNK and SSFB2C are loaded.
- ADABAS, then ADAUSER is loaded, which in turn loads ADARUN, ADAIOR, etc.

Installing Adabas with UTM

This section provides information required for Adabas installation with UTM.

- Operation Options
- Unsynchronized Operation
- Running ADASAV under UTM
- UTM Adalink Entry Points

■ Linking ADAUSER to UTM Applications

Operation Options

UTM can operate with Adabas in two modes:

■ *Synchronized*. UTM transactions and database transactions are coordinated.

UTM is aware of database transactions and coordinates its transactions with Adabas, providing automated restart in case of failure. The selectable unit ADAUTM documented in section *ADAUTM* is required to implement this option.

■ *Unsynchronized*. The application completes database transactions independently of UTM.

UTM is not involved in the database transactions. Several Adabas transactions can occur within a single UTM transaction. Applications call Adabas from the ADALNK module directly. The following sections describe the process of selecting the correct UTM ADALNx entry point.

Unsynchronized Operation

UTM conforms to the KDCS (compatible data communication interface) description, which requires a TP program with the following general sequence:

1. initialize

2. obtain the terminal input data

3. process the data (including any Adabas calls)

4. write the output data to the terminal

5. end (PEND).

Under UTM, a BS2000 task processes only a single user until PEND. By defining multiple UTM tasks for an application, requests are processed in parallel, thus improving performance.

Multiple tasks are also required to prevent deadlock situations. For example, if only one UTM task is available and user 2 requests a record held by user 1, all processing stops (deadlock) until user 2 is timed-out, thus freeing the UTM task to complete user 1's transaction.

Running ADASAV under UTM

The Adabas utility ADASAV should only be run on a UTM system when very few update trans-actions are active, or deadlock can result. This occurs when fewer UTM tasks have been defined than can accommodate the number of Adabas user transactions currently open, and the ADASAV synchronization begins.

When ADASAV performs synchronization, all `ET` transactions are held in the command queue (CQ) until none remain to be processed. If there are more open UTM transactions than UTM tasks available, other transactions cannot be completed until the UTM transactions end; this results in deadlock.

All transactions remain frozen, waiting for time-out to occur. When the first time-out of a UTM transaction occurs, a UTM task is freed and another UTM transaction takes its place. This continues until all frozen transactions are freed, bringing synchronization to an end.

UTM Adalink Entry Points

Software AG strongly recommends that you link ADAUSER to the UTM application and that you use the ADABAS entry point.

For compatibility reasons, ADALNK may be linked to a UTM application using one of the following entry points:

- ADALNN for systems running with Natural, and
- ADALNU for systems running without Natural.

Both are described as provided on the Adabas/BS2000 release tape; however, ADALNN and ADALNU defaults can be changed (zapped). While the Adabas call is being processed, the UTM task waits synchronously until the Adabas nucleus returns the call results.

ADALNN

ADALNN is used where the calling program works with different users, or must identify more than one transaction. However, only one Adabas call is processed at a time, and waits are synchron-ous. ADALNN is used with Natural/UTM or Natural/TIAM/MULTIPASS. The CSECT name is ADALNN. There are no additional ENTRYs.

ADALNU

ADALNU is for UTM applications that do not run with Natural. The UID part of the communication ID is set with the logical terminal ID derived from the KB header field KCLOGTER. For UTM version 3, the KB is found by the weak external KDCKB that is satisfied in KDCROOT. For older UTM versions, or in environments where ADALNU is not linked statically to KDCROOT, ADALNU locates the KB in the program's parameter list by going back up the save area chain to the KDCROOT save area.

Linking ADAUSER to UTM Applications

- Natural
- COBOL or Assembler
- Loading ADALNK

Natural

Software AG recommends that you link ADAUSER to the Natural UTM application and set the Natural driver parameter.

```
ADACALL=NO
```

ADAUSER detects that UTM is the carrier system. It loads ADALNK directly and thus reads its parameters from DDLNKPAR.

COBOL or Assembler

If ADAUSER is linked to a COBOL or Assembler UTM application, ADALNK is loaded and parameters are read from DDLNKPAR.

Loading ADALNK

Whenever you link ADAUSER to UTM applications, you need to add the following link statement to the UTM start job to load the ADALNK:

```
/SET-FILE-LINK DDLIB,<adabas_library>
```

where *adabas_library* is the name of the delivered Adabas module library.

5 Device And File Considerations

Support for new device types that include user-defined block sizes can be implemented in Adabas by modifying one of the table of device-constant entries (TDCEs) reserved for this purpose. A TDCE is X'40' bytes long and the first free TDCE can be identified by X'0000' in its first two bytes (TDCDT).

Under BS2000, the address of the first TDCE is at offset ADAIOR+ X'34' for all versions of Adabas.

Adabas direct access files are always mapped to UPAM files, removing the need to consider physical device characteristics. PAM pages in a file are addressed relative to the beginning of the file.

Adabas blocks comprise one or more PAM pages. An Adabas virtual track is made up of a fixed number of blocks, and an Adabas virtual cylinder comprises a fixed number of tracks. The definition of tracks and cylinders are independent of the physical device.

There are a number of predefined virtual devices for BS2000 that should meet most of the storage capacity needs that arise. It should be noted that the virtual memory requirement increases significantly with a larger block size.

Support for new device types, including user-defined block sizes, can be implemented in ADAIOR by modifying one of the TDCEs reserved for this purpose.

> **Note:** The default device type for BS2000 in Adabas 8.2 is 2300. This device type is compatible with NK4 disks and has 4K rounded blocks for use with D3475-8F devices. The Adabas 8.1 default was 2002; the default 2000 of previous versions will not fulfill both of these conditions.

Information to be Zapped into the First Free TDCE

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

Label	Offset	Contents
TDCDT	00	Device type in unsigned decimal (X'3385'), must be numeric, and unique among all TDCEs
TDCKSN	02	Constant set number: must be uniquely chosen from the values X'28' (reserved for BS2000 device type 2006), X'2B', or X'2E'
TDCF	03	The flag bit must be set—TDCFFBA (X'80') for FBA/PAM devices or TDCFCKD (X'40') for CKD devices
TDCDT1	04	Set to zero under BS2000
TDCDT2	05	Set to zero under BS2000
TDCDT3	06	Set to zero under BS2000
TDCDT4	07	Set to zero under BS2000

Label	Offset	Contents
TDCMSBS	08	In BS2000, 32760 for compatibility. Refer to the TDCMSBS default table in *Maximum Sequential Block Size* in the Adabas z/OS installation instructions for more system- and device-related information.
TDCTPC	0A	Number of tracks per cylinder
TDCCIPT	0C	Number of FBA blocks or PAM pages per track (if TDCFFBA is set). For BS2000 less than or equal to 16.
TDCBPCI	0E	Number of bytes per FBA block or PAM page (2048 if TDCFFBA is set)
TDCABPT	10	Number of Associator blocks per track
TDCABS	12	Associator block size
TDCACPB	14	Number of FBA blocks or PAM pages per Associator block (if TDCFFBA is set)
TDCDBPT	16	Number of Data Storage blocks per track
TDCDBS	18	Data Storage block size
TDCDCPB	1A	Number of FBA blocks or PAM pages per Data Storage block (if TDCFFBA is set)
TDCWBPT	1C	Number of Work blocks per track
TDCWBS	1E	Work block size
TDCWCPB	20	Number of FBA blocks or PAM pages per Work block (if TDCFFBA is set)
TDCTSBPT	22	Number of TEMP or SORT blocks per track (if TDCFFBA is set)
TDCTSBS	24	TEMP or SORT block size
TDCTSCPB	26	Number of FBA blocks or PAM pages per TEMP or SORT block (if TDCFFBA is set)
TDCPBPT	28	Number of PLOG blocks per track
TDCPBS	2A	PLOG block size
TDCPCPB	2C	Number of FBA blocks or PAM pages per PLOG block (if TDCFFBA is set)
TDCCBPT	2E	Number of CLOG blocks per track
TDCCBS	30	CLOG block size
TDCCCPB2	32	Number of FBA blocks or PAM pages per CLOG block (if TDCFFBA is set)

In addition, the sequential protection log block size may have to be increased in the corresponding PTT entry in CSECT I_PTT of the load module ADAIOI. PTT entries begin at offset 0 into CSECT I_PTT.

Each PTT entry is X'10' bytes long and has the structure shown below:

Label	Offset	Contents
PTTPN	00	Program number
PTTFT	01	File type
PTTN	02	DD name characters 2 - 8

Label	Offset	Contents

PTTF	08	Flags:
		
		 ```OUT (X'80') output```
		```BSAM (X'40') BSAM```
		```BACK (X'20') read backwards```
		```JCL (X'10') BLKSIZE/LRECL/RECFM taken from DATADEF statement or label```
		```UNDEF (X'04') undefined record format```
		```VAR (X'02') variable record format```

-	09	Reserved
PTTMBS	0A	Maximum block size
-	0C	Reserved

The PTT entry for the sequential protection log can be identified by X'12F1' in its first two bytes.

General Rules for Defining Device Block Sizes

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

- all block sizes must be multiples of the number of PAM blocks per track (see *Device Types and Block Sizes*, elsewhere in this section)

- a single block cannot be split between tracks (block size must be less than or equal to the track size)

Block Rules for ASSO/DATA

The following rules apply for Associator and Data Storage blocks:

- Associator block size must be greater than one-fourth the size of the largest FDT, and should be large enough to accept definitions in the various administrative blocks (RABN 1 - 30) and in the FCB

- The block sizes for Associator and Data Storage should be a multiple of 256, less four bytes (for example, 1020) to save Adabas buffer pool space

- The Associator and Data Storage block sizes must be at least 32 less than the sequential block size

- Data Storage block size must be greater than: (maximum compressed record length + 10 + padding bytes)

Block Rule for WORK

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

Block Rules for TEMP/SORT

If ADAM direct addressing is used:

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

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

Block size for TEMP and SORT must be greater than the block size for Data Storage.

Block Rule for PLOG or SIBA

The following rules apply for PLOG or SIBA blocks:

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

- It is also recommended that PLOG/SIBA be defined larger than the largest Data Storage block size. This avoids increased I/O caused by splitting Data Storage blocks during online ADASAV operations.

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

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

> **Note:** QBLKSIZE is an ADARUN parameter.

Using 3480/3490 Tape Cartridge Compression (IDRC)

The use of hardware compression (IDRC) is not recommended for protection log files.

BS2000 Device Types and Block Sizes

The primary access method for direct access data sets used by Adabas under BS2000 is PAM (primary access method). The device types defined by Adabas establish a logical structure on a PAM data set in order to process a fixed number of consecutive PAM blocks. For example, for device 2300, one ASSO RABN (Adabas block) consists of two PAM blocks and one logical track consisting of four RABNs.

These device types are "artificial"; there is no relation to the physical devices being used. A maximum of 16 PAM blocks per track can be combined into one I/O call. For more than 16 PAM blocks per track, parameter chaining is used.

The artificial device types defined by Software AG for BS2000 systems are summarized in the following table. The ASSO, DATA, WORK, PLOG, CLOG, and TEMP/SORT/DSIM block sizes are given with PAMs/RABN and RABNs per track in the format:

`PamsPerRabn/blocksize:RabnsPerTrack`

Device	PAM Pages per Cyl[1]	ASSO	DATA	WORK	PLOG	CLOG	TEMP/SORT/DSIM	NK4 Compatible[2]	4K Rounded[3]
2000	80	1/2048:4	2/4080:2	2/4096:2	2/4096:2	2/4096:2	2/4080:2	No	No
2001	152	1/2044:8	2/4092:4	2/4096:4	2/4096:4	4/8192:2	4/8192:2	No	No
2002	152	2/4092:4	4/8188:2	4/8192:2	4/8192:2	8/16384:1	8/16384:1	Yes	No
2003	255	1/2044:15	3/6140:5	3/6144:5	3/6144:5	5/10240:3	5/10240:3	No	No
2004	255	3/6140:5	5/10236:3	5/10240:3	5/10240:3	15/30720:1	15/30720:1	No	No
2005	220	1/2044:20	2/4092:10	4/8192:5	4/8192:5	5/10240:4	5/10240:4	No	No
2006	220	2/4092:10	4/8188:5	5/10240:4	5/10240:4	5/10240:4	5/10240:4	No	No
2007[4]	255	5/10236:3	15/30716:3	15/30720:3	15/30720:3	16/30720:3	16/30720:3	No	No
2008	272	2/4092:8	16/32656:1	16/32760:1	16/32760:1	16/32760:1	16/32760:1	Yes	No
2009	272	2/4092:8	16/32656:1	16/32740:1	16/32740:1	16/32740:1	16/32740:1	Yes	No
2010	240	2/4092:8	4/8188:4	8/16380:2	8/16380:2	8/16380:2	8/16380:2	Yes	No
2200	240	2/4092:8	4/8088:4	8/16380:2	8/16380:2	8/16380:2	8/16380:2	Yes	No
2201	180	2/4092:6	6/12184:2	6/12288:2	6/12288:2	6/12288:2	6/12288:2	Yes	No
2202	240	2/4092:8	8/16280:2	8/16380:2	8/16380:2	8/16380:2	8/16380:2	Yes	No
2300	152	2/4096:4	4/8192:2	4/8192:2	4/8192:2	8/16384:1	8/16384:1	Yes	Yes

Device	PAM Pages per Cyl[1]	ASSO	DATA	WORK	PLOG	CLOG	TEMP/SORT/DSIM	NK4 Compatible[2]	4K Rounded[3]
2301	240	2/4096:8	8/16384:2	8/16384:2	8/16384:2	8/16384:2	8/16384:2	Yes	Yes
2302	272	2/4096:8	16/32768:1	16/32768:1	16/32768:1	16/32768:1	16/32768:1	Yes	Yes

> **Notes:**

1. The PAM Pages per Cyl (PAM pages per cylinder) column shows how many PAM pages are allocated per cylinder (for example, in an ADAFRM utility where the size does not have the suffix "B").

2. Generally, NK4-compatible containers should have an even number in the RABN STD block size.

3. 4K-rounded devices need RABN block sizes to be an exact multiple of 4K bytes. These are usually used with D3475-8F devices. When such devices are shared with UNIX machines, this is the case. Otherwise, the user can apply the optional zap AY824052 to avoid the error message: ADAAI6Y file: DDASSOR1, Device-Type 8f extension needs DEVICE=23xx

4. Although supported, the 2007 device is not recommended for use with Adabas. Support for the 2007 will be removed in the next Adabas release.

If the current database device is not of a compatible type for NK4 disks and it is necessary to migrate it to those disks, you must use the ADAORD RESTRUCTUREDB utility as described in *Adabas Utilities.*

Container Size Calculation

To calculate the size of a container in PAM pages, take the number of RABNs for the container and multiply it by the RABN STD block size.

To check your container file, multiply the highest RABN from the ADAREP utility by the RABN STD block size, and compare it with the HIGH-US-PA field of the `SHOW-FILE-ATTRIBUTES` `file-name,ALL` command output. If the HIGH-US-PA field is less than the highest PAM page calculated from the ADAREP, you will receive a DMS0922 error from the operating system when accessing this.

Splitting Data Sets Across Volumes

For private volumes, splitting is possible under every LOGON user ID:

In ISP format:

```
/FILE data-set,DEVICE=D3480,VOLUME=PRIV01,SPACE=60000
/FILE data-set,DEVICE=D3480,VOLUME=PRIV02,SPACE=60000
```

In SDF format:

```
/CREATE-FILE data-set,PRIV-DISK(SPACE=(60000),VOLUME=PRIV01)
/MOD-FILE-ATTR data-set,PROT=(USER-ACC=*ALL)
/MOD-FILE-ATTR data-set,SUP=PRIV-DISK(SPACE=(60000),VOLUME=PRIV02)
```

For public volumes, the splitting is possible under every LOGON user ID if the master catalog entry of the pubset has the attribute:

```
PHYSICAL-ALLOCATION=USER-ALLOWED
```

This attribute is set by issuing the following command under TSOS:

```
/MOD-MASTER-CAT CAT-ID=ABC,PHYSICAL-ALLOCATION=USER-ALLOWED
```

Once this attribute is set, it is possible to split a data set across two or more public volumes under any LOGON user ID that has the right of space allocation on that particular pubset.

In ISP format:

```
/FILE data-set,VOLUME=ABC.00,DEVICE=D3480,SPACE=60000
/FILE data-set,VOLUME=ABC.01,DEVICE=D3480,SPACE=60000
```

In SDF format:

```
/CREATE-FILE data-set,PUB(SPACE=(60000),VOLUME=ABC.00)
/MOD-FILE-ATTR data-set,PROT=(USER-ACC=*ALL)
/MOD-FILE-ATTR data-set,SUP=PUB(SPACE=(60000),VOLUME=ABC.01)
```

At this point, even a particular physical allocation can be made.

In ISP format:

```
/FILE data-set,VOLUME=ABC.02,DEVICE=D3480,SPACE=(20002,60000,ABS)
```

In SDF format (following the CREATE-FILE and MOD-FILE_ATTR...PROT specifications listed earlier):

```
/MOD-FILE-ATTR data-set,SUP=PUB(SPACE=ABSOLUTE(20002,60000),VOLUME=ABC.02)
```

The example extent covers physical PAM pages 20002 through 80001 on volume ABC.02. The required disk space must, of course, be available. If you are unsure of the available disk space, consult your system administrator.

Saving the Extent List of Data Sets

The utility ADAR2E converts the extent list of given data sets into a JOB containing /CREATE-FILE commands. For more information, see the section on the ADAR2E utility in the *Adabas Utilities* documentation.

6   **Installing The AOS Demo Version**

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

📄 **Notes:**

1. To install the full version of AOS, read the *Adabas Online System* documentation.

2. Demo versions of Adabas Vista (AVI), Adabas Fastpath (AFP), Adabas SAF Security (AAF), and Adabas Transaction Manager (ATM) are automatically installed when you install either the demo or full version of AOS.

The AOS demo version requires the same Natural version as the corresponding release of Adabas Online System. Please refer to the appropriate Adabas Online System documentation to determine its Natural requirements.

AOS Demo Installation Procedure

▶ **To install the AOS demo version without the System Maintenance Aid:**

1   The AOSASM module must be linked to the shared portion of the Natural nucleus.

2   Optionally, set the AOS defaults. Parameters that control the operation of AOS can be set at installation time by changing the defaults in the Natural program AOSEX1 found in library SYSAOSU. For complete information about these parameters, read *Setting the AOS Demo Version Defaults*, elsewhere in this guide.

3   After setting the AOS defaults in the previous step, copy the AOSEX1 member and its companion member P-AOSEX1 from the SYSAOSU library to the SYSAOS library. The programs for AOS are stored in library SYSAOS, and these members and the correct AOSEX1 parameters for your environment must be present in SYSAOS for AOS to run.

   The SYSAOSU library is provided to ensure that AOS settings (including the AOSEX1 settings) in your running AOS installation are not overwritten when you upgrade or apply maintenance to your AOS code. Whenever you upgrade or apply maintenance, you must ensure that the AOSEX1 member in the SYSAOSU library is updated appropriately and copied (with P-AOSEX1) to the SYSAOS library.

4   Perform a Natural INPL.

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

   The distributed INPL jobs (both the sample jobs and the SMA-generated jobs) that you use to load the Adabas INPL library load it in a date-sensitive manner. In other words, the load

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

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

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

> **Note:** When migrating an Adabas 7.4 installation, this procedure does not apply. Instead, you should replace the 7.4 INPL library members with the latest Adabas 8 INPL library members, regardless of the dates of the members, to avoid creating a library containing members from both releases.

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

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

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

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

Installing AOS with Natural Security

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

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

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

▶ **To set-up AOS using Natural Security:**

1   Define at least the library SYSAOS to Natural Security

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

2   Specify the startup program for SYSAOS as MENU

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

Setting the AOS Demo Version Defaults

Parameters that control the operation of Adabas Online System can be set at installation time by changing the defaults in the Natural program AOSEX1. Once you have altered the parameters as needed for your installation, copy the AOSEX1 and P-AOSEX1 members from the SYSAOSU library to the SYSAOS library.

The table below lists the parameters and possible values.

Parameter	Valid Values	Default	Description
ADMIN-LEVEL	0-9	6	Administration level: Allows access to certain functions that can cause error conditions in the ADABAS environment. When set to 8 or higher, it allows the "CATCH RSP-CODE" direct command to occur, and when set to 9, it allows the "ZAP" function to be issued.
AOS-END-MSG	Yes (Y) or No (N)	Y	Display AOS end-of-session message?
AOS-LOGO	Yes (Y) or No (N)	N	Display AOS logo?
BATCH-ERROR	Yes (Y) or No (N)	N	Batch job cond code: When AOS is executing from a batch job and has an error condition, and if BATCH-ERROR is set to "Y", AOS will terminate with a condition code of 8. This function will be fully implemented over time, as all AOS programs must be modified for this.
BLK-CYL	Cylinder (C) or Block (B)	B	Space control by block or cylinder
CMD-INT	Natural (N) or AOS (A)	A	Pass-through control for invalid AOS commands: "N" passes invalid commands to Natural; "A" displays an error message for invalid commands.
CPEXLIST	No (N) or Yes (Y)	N	Display extended checkpoint list? A value of "N" displays the normal list; a value of "Y" displays the extended list.
EX1-A1	---	---	Reserved for future use.
EX1-N3	---	---	Reserved for future use.
EXF-UTI	E or U	U	UTI or EXF file lock exception. A value of E specifies an EXF exception; a value of U specifies a UTI exception.
MAX-AC-IOS	0-999999	150	AC read converter block threshold value
MAXANZ	1-99999999	100	Maximum displayed user queue elements

Parameter	Valid Values	Default	Description
NR-EXT	1-5	4	Critical extent threshold for listing file. This parameter applies to Adabas 7.4 (or earlier) installations.
NR-EXT2	1-99	50	Critical extent threshold for listing file. This parameter applies to Adabas 8 (or later) installations.
NR-PERCENT	1-99	89	Report function: NR-PERCENT is a threshold value for the display of critical files concerning the percentage full of the extents reached in AC/UI/NI/DS table type. A value greater than NR-PERCENT will be highlighted.
PURGE-UQE	Yes (Y) or No (N)	N	Remove user queue element?
SAVEFDT	Yes (Y) or No (N)	N	Keep deleted file's FDT?
STATINTV	1-9999 seconds	60	Statistics-gathering interval, in seconds
TID-DISPLAY	B, A, I	I	Control display for TID in "display user queue" function: "B" = binary TID display; "A" = alpha TID display; "I" = alpha for A-Z/0-9, otherwise binary.
TIMELA	0-99999999 seconds	0 (no limitations)	Display user queue elements with activity during the last "n" seconds.
TIN-JOBN	T or J	J	Display either job name or time-in in "display command queue" function. A value of "T" indicates that time-in should be displayed; a value of "J" indicates that the job name should be displayed.

To change the defaults, you must edit the Natural AOSEX1 program and make the changes directly within the program listing in the defaults area, which looks as follows:

```
  .
    .
      .
DEFINE DATA PARAMETER USING P-AOSEX1
END-DEFINE
*
* SET THE DEFAULTS
*
ADMIN-LEVEL = '6'    (Allows access to certain functions that can cause error ↵
conditions in the ADABAS environment)
AOS-END-MSG = 'Y'    (Display end-of-session message)
AOS-LOGO = 'Y'       (Adabas Online System logo display-set to 'N' for no logo ↵
display)
BATCH-ERROR = 'N'    (If BATCH-ERROR is set to "Y", AOS will terminate with a ↵
condition code of 8 if an error occurs.)
BLK-CYL  = 'B'       (Space allocation default-set to 'C' for cylinders)
CMD-INT  =  'A'      (Pass invalid Adabas commands to (N)atural, or intercept (A))
CPEXLIST = 'N'       (Checkpoint list control-set to 'Y' for extended checkpoint list)
NR-EXT2 = '50'       (ADA V8 critical extent threshold.  Range: 1-99)
EXF-UTI  = 'U'       (File locking exception-set to 'E' to except files in EXF status)
```

```
MAXANZ  = 100        (Maximum user queue elements displayed. range: 1 - 99,999,999 ↵
elements)
NR-EXT = 4           (ADA V7 critical extent threshold.  Range: 1, 2, 3, 4, or 5)
NR-PERCENT = '89'    (NR-PERCENT is a threshold value for the display of critical ↵
files)
MAX-AC-IOS = 150     (AC read converter block threshold)
PURGE-UQE  = 'N'     (Remove element from user queue. Pre-5.1 default is "Y")
SAVEFDT  = 'N'       (Keep old FDT for SAVE operation-set to 'Y' to save FDTs)
STATINTV = 60        (Statistic-gathering time. range: 1 - 9999)
TID-DISPLAY = 'I'    (TID display control: B=binary, A=alpha, I=normally alpha, ↵
special characters as binary)
TIMELA = 0           (Include activity in last 'n' seconds. range: "all" (0) -last ↵
99,999,999 seconds)
TIN-JOBN = 'J'       (Command queue display-"J" for job name, "T" for "time in ↵
queue" )
*
END
```

7 Installing The Recovery Aid (ADARAI)

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

ADARAI Installation Overview

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

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

ADARAI Installation Procedure

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

▶ **To install the Adabas Recovery Aid:**

1   Define and format the recovery log files

     The DDRLOGR1 and DDRLOGM1 files must be on the same device type.

     Use the ADAFRM RLOGFRM function to format the RLOGs.

     Use the ADAFRM RLOGFRM `MIRROR` parameter to format the DDRLOGM1 file.

2   Add data definition statements for the recovery log files

     Add DDRLOGR1 and DDRLOGM1 DD statements to the nucleus job stream and to any utilities that update or save the database and thus write to the RLOG files.

     Whenever these utilities are executed while ADARAI is active in the database (that is, after the PREPARE function has been executed), the DDRLOGR1 and DDRLOGM1 DD statements must be included.

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

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

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

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

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

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

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

3   Install ADARAI on the database.

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

4   Create the first ADARAI generation.

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

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

8 Adabas Dump Formatting Tool (ADAFDP)

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

ADAFDP Function

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

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

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

ADAFDP Output

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

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

ADAFDP Messages

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

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

Message	Description
ICCB POINTED FROM X'A0' IN IOR	The ICCB address to which this offset in ADAIOR points.
ADAI22 ADAIOR TRACE TABLE	Format of ADAIOR trace table; same as that found with the ADAM99 message.

Pool Abbreviations

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

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

User Threads

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

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

Command Information

Information	Description
CQE Address	The address location of this CQE

F	Command queue flag bytes:
	
	■ First Byte: General Purpose Flag
	
	  ■ X'80': User buffers in service partition, region, address space
	
	  ■ X'40': ET command waiting for 12 call
	
	  ■ X'20': Waiting for 16 call
	
	  ■ X'10': 16 call required
	
	  ■ X'08': Attached buffer
	
	  ■ X'04': Attached buffer required
	
	  ■ X'02': X-memory lock held (z/OS only)
	■ Second Byte: Selection Flag
	  ■ X'80': In process
	
	  ■ X'40': Ready to be selected
	
	  ■ X'20': Search for UQE done
	
	  ■ X'10': UQE found
	
	  ■ X'08': Not selectable during BSS=x'80' status
	
	  ■ X'04': Not selectable during ET-SYNC
	
	  ■ X'02': Waiting for space
	
	  ■ X'01': Waiting for ISN in HQ

CMD	The command type
File Number	The file number for this command
Job Name	Job name for the user
Addr User	UQE Address of users UQE, if searched for and found

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

RABN Information

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

9 ADAUTM (Universal Transaction Monitor Support)

Software AG provides ADAUTM to coordinate Adabas database operations with the SNI BS2000 systems running the Universal Transaction Monitor (UTM).

This section describes ADAUTM function, installation and operation.

Common Synchronization Points

To safely operate SNI's Universal Transaction Monitor (UTM) and a database, both systems must be able to run separately and coordinate their restarts. That is the only way to synchronize a database transaction running inside a UTM transaction.

ADAUTM makes it possible to have a common point of synchronization and thereby to set the whole transaction back to the last common synchronization point, if needed.

Both Adabas and UTM define points of synchronization ("sync points"):

- Adabas defines a sync point at the end of the database transaction;
- UTM defines a sync point at the end of a UTM transaction.

In some cases, these sync points are common to both Adabas and UTM. If an emergency restart should be necessary, both of the systems can set the interrupted transactions back to the last common sync point.

A database connection module (DBCON) is used for communication between UTM, its user program, and Adabas. The use of synchronization mode is controlled by a parameter in the UTM utility KDCDEF.

The UTM Transaction Concept

The UTM transaction concept is to be referenced to the database access here only.

Each UTM user program starts a transaction using an INIT call and ends with a PEND *nn*. When *nn* takes on the value RE or FI, UTM writes a synchronization point followed by an end-of-transaction for the database transaction. Normally, a user program determines the end of a database transaction. In an Adabas environment, the DBCON can take over that task. After an end-of-transaction `ET` for a database, no more calls to this database are allowed within the current UTM transaction.

Comments and Limitations

DBCON is invoked from UTM only if it occurs on a TP/DB transaction. If the last transaction within a UTM process is a pure TP transaction, the start parameter `VG-ENDE=CL` cannot be carried out at the end of the UTM process.

A 16-byte prefix of `ET` data is written at every sync point, in the order of synchronization. This prefix is transparent to the user. This means that when a UTM process contains more than one UTM transaction and the user wants the `ET` data to be available after the end of the process, the `ET` data must be written at every `ET` or with the last `ET/CL` for this DB transaction. Otherwise, the former `ET` data will be overwritten by the prefix.

A UTM process imposes only a few limitations on ADAUTM at this time. Those limitations refer only to the Adabas commands `OP`, `ET`, and `BT`. When using an `OP` command, the R option cannot be used if a process contains more than one UTM transaction because the record buffer is not available after the end of a database transaction. In case of `ET` and `BT`, the ISN hold option cannot be used.

The length of `ET` data that can be used is limited to 1984 bytes for users.

ADAUTM Functions

In the database transaction process, all calls to the database are transferred to DBCON to be checked. The start, update, and end of transaction are the most important parts.

At the start, a transaction is checked to see if the first call is an `OP` command and if so, whether the Additions 1 field of the Adabas control block contains a valid value. If the Additions 1 field does not contain a valid string, a generic one is created and inserted. If the first command is not an `OP` command, such a command is carried out implicitly; after that, the user's order is handed over to Adabas.

All commands that have, either explicitly or implicitly, an attribute update are checked. The database ID of the first commands of that category of a transaction is marked as a update database for this transaction. That means that it is not allowed to carry out updates for more than one database within one database transaction.

Usually, the user initiates the end of transaction by issuing an `ET/CL` command. If that is not the case, the DBCON generates the UTM end of transaction. At this point, calls to the database are not permitted until the end of the UTM transaction (otherwise, the whole transaction must be set back to the last sync point).

When processing a UTM transaction, the main task is to handle the end of transaction procedure. Here, the synchronization with the database is done. In any case, `ET` data written for synchronization

purposes is transparent to the user. Another listed function is the backout of a TP/DB transaction, as well as the functions with special positions, such as those to process status information. That special function is used during emergency restart before UTM is active.

- Establishing and Terminating a Connection
- User Call
- UTM End of Transaction
- UTM Transaction Backout
- Coordinated Restart
- User Exit 1

Establishing and Terminating a Connection

- Connecting to Adabas
- Disconnecting from Adabas

Connecting to Adabas

The initial link may be subdivided into the following parts:

- setting default values that are not defined at session startup, using the appropriate parameters

  All parameters that did not acquire a value during the start of the UTM session are now initialized by a default value. These defaults are described in a later section.

- creating the internal administration tables

  A table is built that contains information about the user and the user's transactions. The table is installed as a common memory pool. The value of the parameter APPLI-ID is used as part of the pool's name.

- making the database and the running mode available

Disconnecting from Adabas

When disconnecting from Adabas, the administration pool is closed in accordance with BS2000 conventions.

User Call

All user calls to Adabas pass through this function. Here, all the necessary checks are carried out. These checks refer to the synchronization of the TP and DB transaction, but not to the syntax of the Adabas commands or their buffers. The functionality is described with an example of a UTM transaction that contains accesses to a database.

At the beginning, DBCON determines whether the current database call is the first within the UTM transaction. If it is the first call, DBCON determines whether it is an `OP` (open) command call to initiate an Adabas transaction. If it is not an `OP` command, an internal `OP` is performed after the user's call is completed.

DBCON also determines whether a command leads or could lead to an update in any form. If so, the database ID is recorded. With the occurrence of the first update-type command, DBCON marks the transaction as an update transaction. From that point, updates may only be done on this database during this transaction. Modifications to any other databases are rejected as errors, and the whole transaction is set back.

At the end of a database transaction, the `ET/CL` (end-of-transaction/close) commands are not passed immediately to Adabas, but instead are delayed until the end of the UTM transaction. This means that between the end of a database transaction and the end of the related UTM transaction, no more calls to the database can be carried out. Any attempt to execute such calls leads to a backout of that transaction. At the end of the UTM transaction, it is determined whether the user has determined the end of the transaction, or if DBCON must end it internally.

- User Open
- Internal Open

User Open

If the field ACBADD1 contains a value equal to zero or blank, DBCON tries to find an `ET` data ID in the administration pool (subsequent transaction within a UTM process). When available, this ID is taken over into ACBADD1; otherwise, an ID is built by the constant ADAU, the APPLI-ID, and an internal number. If a user exit is available, it gets control in order to modify the passed Adabas control block. The only change the exit can make is to modify the value in ACBADD1. After the DBCON gets back control, it overwrites the field with the old value if the exit has changed it to zero or blank, or the `ET` data ID was changed after the first transaction of this process.

Internal Open

If the first database call is not an `OP` command and `ET-MODE=AUTO` (the default) is activated, DBCON determines whether or not this is a subsequent transaction within the current UTM process. If it is, the call will be executed. If Adabas returns the call with a response code 9 (ADARSP009), an internal `OP` command is created and executed; after that, the user's call is repeated. At the beginning of a UTM process, an internal `OP` command is created, carried out, and after that, the user call. The internally generated `OP` command is passed to the user exit in both cases. See also the section *User Exit 1*.

UTM End of Transaction

The function PEND with one of the attributes `RE,` `FI,` or `FC` determines the end of transaction of UTM. The TP transaction end also contains an end of a DB transaction. That means that now the commands `ET/CL` are carried out. These were delivered either by the user, or must be created by the DBCON now. `ET/CL` commands always get an "E" option in the Adabas control block. Before Adabas is called, the DBCON passes the command through to the exit. After a successful execution of the command with a PEND-RE, data about this transaction is stored in the administration pool.

UTM Transaction Backout

This function is called by UTM if a user executes a RESET or PEND-ER call. The DBCON calls this for itself if the database is not available or an error occurs during the end of transaction procedure. Adabas return codes that occur during a transaction are passed directly back to the user and do not lead to a backout of the transaction.

Coordinated Restart

The coordinated restart contains two functions which are called during the phase of emergency restart and/or the normal end of the application. An emergency restart indicates an abnormal end of the last UTM session. At a new start of the application, one of two conditions may occur:

- there are open transactions
- there are no open transactions

While the UTM transaction is still open, it must be determined whether or not the database transaction is also still open. This occurs as described below.

In the case of emergency restart, the DBCON is called by the order Check DB Status for update transaction. At first, the `ET` data belonging to the update database are read. Then all read-only database operations are handled in the same way. The following situations may occur:

- if the synchronization data of UTM are equal to that read from the update database `ET` data header, the transaction is marked as finished for UTM.

- if the synchronization data of UTM are not equal, the complete DB transaction is backed out (BT) and UTM is informed with "transaction canceled". If Adabas returns a response code unequal to 9 (ADARSP009) or 22 (ADARSP022), the process is stopped. The UTM utility KDCDEF may be required.

If there are no open transactions, UTM requires the DBCON to delete all information needed for synchronization. In addition, after a normal end of the last UTM session, the DBCON is required to delete all information.

User Exit 1

User Exit 1 can be used to influence the Adabas commands OP, ET, and CL.

However, User Exit 1 can only modify the commands which are offered; it cannot execute its own Adabas command because control is not passed back after executing such a command.

The Adabas control block and the record buffer may be modified depending on the commands being used.

The field E1PARB contains the address of the ET data area provided by ADAUTM. The address at E1PARB points to a field which contains the record length and is a part of a 16-byte header. With the exception of the length field, the header cannot be modified. The current length of the new data, plus 16 (the length of the header) must be set in the length field.

- Processing at the Start of a UTM Process
- Processing at the End of a UTM Update Transaction/Process
- User Exit Parameter List

Processing at the Start of a UTM Process

The read ET data option with the OP command cannot be inserted by the exit. The value that can be modified is the ET data ID in the Additions 1 field of the Adabas control block.

The ET data ID can only be modified at the beginning of a UTM process. It cannot be modified after a PEND RE if the DBCON has to issue an OP (after an Adabas error 9).

When using an OP command, only the fields E1PFB and E1PARB can be modified.

The ET data ID in the Additions 1 field can only be modified as follows:

- If bit E1PAR is on, the ET data ID cannot be modified.

- Bit E1POP and E1PIO allow the ET data ID to be modified; however, they cannot be changed to either zero or blank.

Processing at the End of a UTM Update Transaction/Process

An update transaction at the end of a UTM transaction or process can exchange `ET/CL` commands with each other and/or make `ET` data available.

At the end of a UTM transaction/process, the following conventions apply:

- User Exit 1 gets control only for an update transaction

- the contents of the Adabas command field may only be modified by the exit according to the start parameter `VG-ENDE`

- checks must not occur after returning from the exit

- the `ET` data ID may not be modified because its range of validity is the UTM process. `ET` data itself may be changed; the exit must provide the new data in its own area.

- the current length must be delivered in the field Length of the record buffer in the Adabas control block

If the exit sets the bit E1PFBRE, ADAUTM moves the data into its own area; otherwise, no user `ET` data are written.

A maximum of 1984 bytes of user `ET` data are allowed.

If there was an internal `ET/CL`, the field E1PARB contains low-value.

User Exit Parameter List

The following description contains the action fields of the parameter list. The complete list can be taken from the Assembler DSECT EX1PARM.

```
E1PTAS1
E1POP: User with its own OPEN command.
E1PAR: If the first command is not an OPEN after PEND RE, it is done after response ↵
code 9 (ADARSP009).
E1PIO: It is provided an internal OPEN command.
E1PFB
E1PFBCO : EXIT1 has modified the ACB during an OPEN
E1PFBCE : EXIT1 has modified the ACB during an ET/CL command.
E1PFBRE : EXIT1 has modified the RB during an ET/CL command.
E1PACB Address of ACB, either of the user or of ADAUTM.
E1PARB Address of RB; will be delivered by ADAUTM or is equal to zero using an ↵
internal ET/CL.
The exit 1 provides an address of its own record buffer, if an internal ET/CL occurs.
```

The exit is called by the standard BALR interface of the Assembler language. The following registers are used:

```
R01 : Address of the parameter list
R13 : Address of the save area
R14 : Return address
R15 : Start address of the modules
```

ADAUTM Installation

To make a coordinated restart possible, the DBCON (ADAUTM) must be specified in the UTM root module of the application. This is done by using the DATABASE statement in the UTM utility KDCDEF. The DATABASE statement defines the database in the source of the UTM root. In addition, the ADAUTM modules and the Adabas link module must be provided for the linkage editor's run.

- Installation Procedure
- ADAUTM Parameters
- ADAUTM Parameter Example

Installation Procedure

The following procedure must be used to install ADAUTM.

▶ **To install ADAUTM:**

1   Run KDCDEF with:

```
DATABASE TYPE=DB,ENTRY=xyz
```

—where `ENTRY=xyz` must match the name used in Adabas calls (usually "ADABAS"). If `ENTRY` is omitted or specified as "DB", `ENTRY=ADABAS` is generated.

The `LIB` parameter of the DATABASE statement does not apply for ADAUTM because ADAUTM must be statically linked with KDCROOT.

2   Assemble KDCROOT with the macro library AUT*nnn*.MAC where *nnn* is the ADAUTM version, revision, and SP level numbers).

3   Link the UTM application with:

- AUTDB*mm* from AUT*nnn*.MOD where *mm* is the TM version (31, 32, 33, 34, and 40 are supported);
- AUTNUC from AUT*nnn*.MOD; and
- ADAUSER from ADA*nnn*.MOD

Usually, KDCROOT will contain an entry point ADABAS for Adabas calls. ADAUSER also has a symbol ADABAS. It is therefore necessary to rename this symbol on binding ADAUSER. In $TSOLINK, specify:

```
RENAME ADABAS,ADABASX
INCLUDE ADAUSER,SAG.ADALIB
```

Alternatively, if using the Binder, specify:

```
INCLUDE-MODULES LIB=&ADA,ELE=ADAUSER,TYP=R
RENAME-SYMBOL SYMBOL-NAME=ADABAS,NEW-NAME=ADABASX
```

4   Add ADAUTM parameters to the UTM startup parameters in the UTM startup procedure.

ADAUTM Parameters

This section describes the ADAUTM parameters.

ADAUTM Parameter Syntax

The syntax of an ADAUTM parameter is

```
.DB ADABAS parameter = value
```

If the prefix ".DB ADABAS" is omitted, UTM sends the message "K38 -Parameter Error-". The application ends abnormally.

ADAUTM Parameters

Parameter	Description	Possible Values	Default
DATABASE DA DB	Default database ID that is the update database for the current session.	1 - 65536	1
APPLI-ID AID	Required. The short name of the UTM application. The application ID identifies the UTM application running against Adabas. This value becomes part of the name of the common memory pool. That means that, in accordance with the parameter SCOPE, a default value may lead to an error of the UTM process administration.	1 - 9999	1
ET-MODE ETM	ADAUTM generates the OP, ET, CL commands if the application does not do it. ETM=AUTO will cause an OP command to be generated at the beginning of a database transaction, and an ET/CL command is automatically executed at the end of a UTM or database	AUTO | MAN	AUTO

Parameter	Description	Possible Values	Default
	transaction, when necessary. Both are required for synchronization. `ETM=MAN` disables this function.		
VG-ENDE VGE	Generate a `CL` instead of an `ET` command at the end of the UTM process. `VGE=CL` generates a `CL` command for the Adabas transaction if the UTM transaction finishes with PEND FI/FC; otherwise, an `ET` is generated.	CL | ET	ET
UEX1	The name of the user exit that gets control during `OP/ET/CL` commands. Name of the module that is linked from a library with the link name DDLIB.	module name	(none)
SCOPE	Accessibility of the common memory pool for ADAUTM. `SCOPE` applies only for TASKTYPE "BATCHTASK" and "TP". If TASKTYPE is "INTERACTIV", the scope of the common memory pool is always "TASK".	USERID | SYSTEM | TASK | USER_GROUP	USERID
UID-ADA	How the last 8 bytes of the Adabas communication ID are built by ADAUTM. `UID-ADA=KCBENID`: the KB field KCBENID is used `UID-ADA=KCLOGTER`: the KB field KCLOGTER is used `UID-ADA=VGNR`: the rightmost 4-bytes are built using the UTM conversation ID, as in releases of Adabas prior to 6.1, and the leftmost 4 bytes (the prefix) are specified by the `UID-PRF` parameter.	KCBENID | KCLOGTER | VGNR	VGNR
UID-PRF	If the `UID-ADA` parameter value is "VGNR", this parameter specifies the leftmost 4 bytes.	abcd	(none)

ADAUTM Parameter Example

The following is an example of ADAUTM parameter usage:

```
.DB ADABAS DB = 002 , AID = 80
.DB ADABAS VG-ENDE = CL , SCOPE = USERID
.DB ADABAS ET-MODE = AUTO
.DB ADABAS UEX1 = ADAEX1
```

ADAUTM Diagnostic Information

ADAUTM diagnostic information is written to UTM's DB-DIAGAREA and to SYSOUT.

- UTM DB-DIAGAREA
- Messages to SYSOUT
- ADAUTM Message Codes

UTM DB-DIAGAREA

The general layout and use of the DB-DIAG-AREA area are described in the appropriate UTM manual. The DB-DIAGAREA is written as follows:

- *primary*: DB trace information provides essential information in an ADAUTM response code
- *secondary*: DB trace information includes more detail; see the DSECT DDBTRAC generated by the macro DDBTRAC for the layout of the information

Messages to SYSOUT

ADAUTM writes messages to SYSOUT in the format:

```
AUTxxxx date time OP=yyyy UID=abcdefgh DBID=nnnnn RSP=mmm
```

where

```
xxxx is the ADAUTM message code
yyyy is the UTM primary opcode; see the UTM macro DBCONPAA
abcdefgh is the last 8 bytes of the Adabas communication ID
nnnnn is the Adabas database ID
mmm is the Adabas nucleus response code
```

ADAUTM Message Codes

This section describes the ADAUTM messages.

- ADAUTM Message Format
- ADAUTM Messages

■ Handling Adabas Nucleus Response Codes

ADAUTM Message Format

ADAUTM message codes are 4 characters long in the following format:

xnnn

where

x	is the type identifier. Possible values are: ■ D (database): ADAUTM received a response code *nnn* from Adabas. With the exception of D148, Adabas response codes starting with D are not reported. ■ I (internal): an error occurred in ADAUTM's handling of ET data ■ P (parameter): an error occurred in ADAUTM's startup parameters ■ S (system): ADAUTM received an error from a BS2000 executive macro; or, the installation is not correct ■ U (user): ADAUTM received unsupported Adabas calls
nnn	is the Adabas response code received by ADAUTM; see value D above.

ADAUTM Messages

The following table contains the ADAUTM response codes and messages:

Response Code	Message / Description
I100	Internal area for ET data exhausted
P100	Statement format invalid
P101	Unknown parameter
P102	Invalid continuation
P103	Prefix ".DB ADABAS" not correct
P104	Invalid length of statement
P105	Invalid value
P120	Value not numeric
P121	Numeric value out of range
S100	ENAMP error AUTPOOL
S101	REQMP error AUTPOOL
S102	DISMP error AUTPOOL
S106	ESQUTM not linked AUTCONC
S107	TRACE open error AUTCONC

Response Code	Message / Description
S108	ENQAR error AUTPOOL
S109	DEQAR error AUTPOOL
S110	ADALNQ not linked AUTMAIN
S111	Unsupported BS2000 version AUTMAIN
S112	Unsupported HSI version AUTMAIN
S113	HSITYPE error AUTMAIN
S114	TABLE error AUTMAIN
U100	More than four (4) DBIDs used in a single transaction.
U101	Update command issued between ET and end of UTM transaction.
U102	OP command issued, but ET or CL required.
U103	More than one update DBID used in a single transaction.

Handling Adabas Nucleus Response Codes

When ADAUTM receives response code 148 (ADARSP148) from the Adabas nucleus, a "DBMS down" condition is reported to UTM.

10   Translation Tables

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

Adabas EBCDIC to ASCII and ASCII to EBCDIC

```
cUES2ASC DS 0F
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c DC x'000102033F093F7F3F3F3F0B0C0D0E0F'  0.
c DC x'101112133F3F083F18193F3F3F1D3F1F'  1.
c DC x'3F3F1C3F3F0A171B3F3F3F3F3F050607'  2.
c DC x'3F3F163F3F1E3F043F3F3F3F14153F1A'  3.
c DC x'203F3F3F3F3F3F3F3F3F3F2E3C282B3F'  4.
c DC x'263F3F3F3F3F3F3F3F3F21242A293B5E'  5.
c DC x'2D2F3F3F3F3F3F3F3F3F7C2C255F3E3F'  6.
c DC x'3F3F3F3F3F3F3F3F3F603A2340273D22'  7.
c DC x'3F61626364656667686093F3F3F3F3F3F'  8.
c DC x'3F6A6B6C6D6E6F7071723F3F3F3F3F3F'  9.
c DC x'3F7E737475767778797A3F3F3F5B3F3F'  A.
c DC x'3F3F3F3F3F3F3F3F3F3F3F3F3F5D3F3F'  B.
c DC x'7B4142434445464748493F3F3F3F3F3F'  C.
c DC x'7D4A4B4C4D4E4F5051523F3F3F3F3F3F'  D.
c DC x'5C3F535455565758595A3F3F3F3F3F3F'  E.
c DC x'3031323334353637383939F3F3F3F3F3F'  F.
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END
cUES2EBC DS 0F
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c DC x'00010203372D2E2F1605250B0C0D0E0F'  0.
c DC x'101112133C3D322618193F27221D351F'  1.
c DC x'405A7F7B5B6C507D4D5D5C4E6B604B61'  2.
c DC x'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F'  3.
c DC x'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'  4.
c DC x'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D'  5.
c DC x'79818283848586878889919293949596'  6.
c DC x'979899A2A3A4A5A6A7A8A9C06AD0A107'  7.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'  8.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'  9.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'  A.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'  B.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'  C.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'  D.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'  E.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'  F.
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END
```

Entire Net-Work EBCDIC to ASCII and ASCII to EBCDIC

```
NW2ASC DS 0F
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
DC X'000102030405060708090A0B0C0D0E0F' 0.
DC X'101112131415161718191A1B1C1D1E1F' 1.
DC X'00000000000000000000000000000000' 2.
DC X'00000000000000000000000000000000' 3.
DC X'200000000000000000005B2E3C282B5D' 4.
DC X'26000000000000000000021242A293B5E' 5.
DC X'2D2F000000000000000007C2C255F3E3F' 6.
DC X'000000000000000000603A2340273D22' 7.
DC X'006162636465666768690000000000000' 8.
DC X'006A6B6C6D6E6F707172000000000000' 9.
DC X'007E737475767778797A00005B000000' A.
DC X'000000000000000000000000005D0000' B.
DC X'7B4142434445464748490000000000000' C.
DC X'7D4A4B4C4D4E4F505152000000000000' D.
DC X'5C7E535455565758595A000000000000' E.
DC X'303132333435363738397C00000000FF' F.
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
NW2EBC DS 0F
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
DC X'000102030405060708090A0B0C0D0E0F' 0.
DC X'101112131415161718191A1B1C1D1E1F' 1.
DC X'405A7F7B5B6C507D4D5D5C4E6B604B61' 2.
DC X'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F' 3.
DC X'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' 4.
DC X'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D' 5.
DC X'798182838485868788899192939495996' 6.
DC X'979899A2A3A4A5A6A7A8A9C06AD0A100' 7.
DC X'00000000000000000000000000000000' 8.
DC X'00000000000000000000000000000000' 9.
DC X'00000000000000000000000000000000' A.
DC X'00000000000000000000000000000000' B.
DC X'00000000000000000000000000000000' C.
DC X'00000000000000000000000000000000' D.
DC X'00000000000000000000000000000000' E.
DC X'000000000000000000000000000000FF' F.
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END
```

Index