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
- Installing Adabas with Batch / TIAM
- Installing Adabas with 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

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.

Parameter Syntax	Description
DBID= {*nnnnn* | 1 | (*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= (*ADAccccc* | 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 | NO }	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.

Parameter Syntax	Description
GROUPS={ YES | NO | (*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={ ADABAS5B | *ADAccccc* | (*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.
X48={ YES | NO }	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*.

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

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*.

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 ADALN*x* 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 transactions 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 synchronous. 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

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.