Installing Adabas With TP Monitors

This section provides information needed to install Adabas in batch mode and with teleprocessing (TP) monitors Com-plete, CICS and Shadow. Information about using Adabas with TP monitors is also contained in the section describing Adabas installation.

TP Monitor	TP Monitor / Adalink
All environments	All
CICS command-level	LNKOLSC / LNKOLM
CICS high-performance stub	LNCSTUB
Com-plete	ADALCO
Shadow	ADALNS
Batch	ADALNK, ADALNKR(LNKVSER)

- Preparing Adabas Link Routines for z/VSE
- Installing Adabas with CICS
- Installing the CICS High-Performance Stub Routine
- Installing Adabas with Com-plete
- Installing Adabas with Shadow
- Installing Adabas with Batch / TSO

Preparing Adabas Link Routines for z/VSE

- High-Level Assembler
- Addressing Mode Assembly Directives
- UES-Enabled Link Routines

High-Level Assembler

To maintain compatibility with year 2000 date format requirements, use the IBM high-level assembler when assembling the Adabas link routines for TP monitors on z/VSE. The high-level assembler generates 4-digit year assembly dates into the load modules using the &SYSDATC assembly variable.

It is possible to assemble the link routines with the old VSE assembler after editing the source members and making the following changes:

- For all link members

1. Remove the keyword HLASM=YES from the ASMDATE macro in the source.
2. Supply the following statement on the assembly step to provide the assembly date:

   ```
   // OPTION SYSPARM='yyyymmdd'
   —where
   yyyy is the 4-digit year
   mm is the 2-digit month
   dd is the 2-digit day
   ```

   Notes:

   1. The ASMDATE macro can also accept a 2-digit year from SYSPARM (// OPTION SYSPARM='*yymmdd*')
   2. The ASMDATE.E book must be available in the LIBDEF search chain to assemble the link routines with the old VSE assembler.

3. Remove or comment out the AMODE and RMODE directives in the source members.

- For the ADALCO (Com-plete) link routine

  Ensure that the BASSM.E and BSM.E books are in the LIBDEF search chain of the old VSE assembler job step.

Addressing Mode Assembly Directives

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

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

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

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

Software AG recommends RMODE 24 for the z/VSE batch link routine.

For the CICS command-level link routines (modules LNKOLM, LNKOLSC), the directives are

```
ADABAS AMODE 31
ADABAS RMODE ANY
```

Modifying the Assembly Directives

These directives may be changed by modifying the source members before assembling them, or they may be overridden by linkage editor control statements. For example, to link the batch/TSO ADALNK module with AMODE 31 and an RMODE ANY, the following control statements may be provided as input to the linkage editor:

```
MODE AMODE(31),RMODE(ANY)
ENTRY ADABAS
```

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

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

UES-Enabled Link Routines

For Adabas Version 7.4, UES is enabled by default for the batch/TSO, Com-plete, and IMS link routines. It is not necessary to disable UES support. Applications that do not require UES translation continue to work properly even when the UES components are linked with the Adabas link routines. See the section *Connecting UES-Enabled Databases* for more information.

Disabling UES Support

However, if for some reason you feel it necessary to disable UES support in the Adabas link routines, use the following procedure to do so:

1. Edit the source member ADALCO, ADALNI, ADALNK, or ADALNKR. Set the &UES Boolean assembler variable to 0 by commenting out the source line where it is set to 1 and removing the comment from the line where it is set to 0.

2. Assemble the link routine after making any other necessary modifications to the equates and other directives in the source module as required by your installation.

3. Link the Adabas link routine and do not include any of the UES components (that is, LNKUES, ASC2EBC, or EBC2ASC).

Installing Adabas with CICS

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

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

- The command-level link can be used with VSE CICS/VS 2.2 and 2.3.

- CICS TS 1.1 running under z/VSE 2.4 and above must run a current version of Adabas and use the command-level link component.

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

The following sections describes specific points of Adabas/CICS installation and operation from the CICS perspective, depending on the CICS level being used.

- CICS MRO Environment Requirements

- Standard Versus Enhanced Installation
- Fully Reentrant Operation
- DISPGWA Module : Displaying the CICS Global Work Areas
- LNKENAB and LNKTRUE Modules
- JCL and Source Members
- Sample Resource Definitions
- Modifying Source Member Defaults (ADAGSET Macro)
- Installation Procedure

CICS MRO Environment Requirements

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

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

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

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

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

Standard Versus Enhanced Installation

All supported versions of the command-level link routine can be installed using the standard installation, which comprises steps 1 through 3 of the installation procedure.

Steps 1 through 5 are required in order to use the enhanced features of the command-level link routine:

- CICS transaction isolation;
- a fully reentrant command-level link.

Step 6 is used to install the optional DISPGWA program. The DISPGWA program is only available with the enhanced installation.

The enhanced installation is required if

- CICS transaction isolation is used;

- the DISPGWA storage display program is used.

CICS Transaction Isolation

The enhanced Adabas CICS command-level link components take advantage of the transaction isolation facility provided by the CICS transaction server (TS) Version 1.1 when running with specific hardware under z/VSE 2.4 or above.

Transaction isolation is an extension of the storage protection mechanism, which permits resources to access either CICS or user storage by using storage protection keys. Resources defined to operate in

- user key may not overwrite CICS storage, thus affording a degree of protection to CICS.
- CICS key may read or write either CICS or user key storage, affording the highest degree of access to CICS resources.

Using the transaction isolation facility of CICS TS 1.1 under z/VSE 2.4 or above, CICS resources are further protected by isolating them in *subspaces*. This protects user key resources from one another, and protects CICS key resources from the CICS kernel.

Transaction isolation can be enabled globally through the TRANISO system initialization (SIT) parameter, and for each CICS transaction with the new resource definition ISOLATE keyword.

Transaction isolation places some restrictions on CICS resources that must be available both during the life of the CICS system and to all transactions running in the CICS system.

The Adabas CICS command-level link components must be defined to CICS with the proper storage access to ensure proper operation when transaction isolation is active:

- The Adabas CICS command-level link routine, comprising the LNKOLSC, LNKOLM, and CICS entry and exit code, must be defined to CICS as a *user key* program.
- The LNKTRUE and LNKENAB (ADATRUE and ADAENAB) programs must both be defined as *CICS key* programs.

This permits the correct degree of isolation between applications invoking the link routine, and the Adabas CICS task-related user exit (TRUE), which interacts directly with CICS resources.

When CICS transaction isolation is active, user SVCs cannot execute from the CICS region. SVCs can be executed in CICS key, however, during the PLT phase of the CICS startup and termination. For this reason, the module LNKENAB is provided to execute during the PLTPI phase of CICS initialization.

Fully Reentrant Operation

The Adabas command-level link routine can now be made fully reentrant and not self-modifying when the module LNKENAB is executed. During execution, a CICS global work area (GWA) is obtained and passed to the command-level link, which uses the GWA to store initialization addresses. This feature is available for CICS TS 1.1 under z/VSE 2.4 and above.

DISPGWA Module : Displaying the CICS Global Work Areas

The DISPGWA module is a program provided by Software AG as an optional feature for CICS TS 1.1 under z/VSE Version 2.4 and above.

The DISPGWA program displays the global work area (GWA) used by the various command-level link components. It can be used to display other areas of CICS that are important to the Adabas command-level link routine when it is executing as a task-related user exit (TRUE). With the help of Software AG personnel, you can use this program to interrogate important data areas during problem determination.

The DISPGWA module is used only if the LNKTRUE module is used, since it is the LNKTRUE module that actually EXTRACTs the global work area.

LNKENAB and LNKTRUE Modules

This section describes the usage of the LNKENAB and LNKTRUE modules.

LNKENAB Module

The LNKENAB module

- starts and enables the task-related user exit LNKTRUE (see the next section) during PLTPI processing.
- defines the length of storage that CICS gives to LNKTRUE as each task is invoked for the first time. The storage remains in CICS until the task terminates and is used by LNKTRUE as a task work area.
- issues an Adabas command to the default target and Adabas SVC defined in the ADAGSET macro in LNKOLSC. The target need not be active.

The purpose of the command is to derive the address of the IDTH from the first SVC call. All other SVC calls from the command-level link routine are then made using a branch entry into the Adabas SVC.

LNKTRUE Module

When started and enabled by LNKENAB during PLTPI processing, the task-related user exit LNKTRUE module

- permits the command-level link routine to obtain the pointer to the access control environment element (ACEE) in a CICS/ESA 4.1 environment;
- facilitates processing when CICS transaction isolation is installed and enabled; and
- coordinates Adabas transactions through the CICS Resource Manager Interface (RMI) when the Adabas Transaction Manager (ATM) is installed and enabled.

Most of the command-level link components execute under the umbrella of LNKTRUE. This means that any abend condition during the execution of LNKTRUE is serious; CICS may even respond by terminating the entire CICS region.

Items that may cause this condition include, but are not limited to

- invalid application parameter lists for Adabas calls;
- inconsistent or incorrect keyword values coded in the ADAGSET macro for the various command-level components;
- user modification to the command-level link components or the task-related user exit; or
- incorrect coding in UEXITA and/or UEXITB components.

Software AG strongly recommends that you test application programs in a CICS region that supports the non-task-related user exit version (standard installation) of the command-level link before migrating to a task-related user exit version (enhanced installation) of the command-level link routine.

JCL and Source Members

The JCL members use the sources indicated in the following table:

JCL	Source	Source Description
CICSASMC	LNKOLSC/LNKOLM	LNKOLSC is the dependent part of the Adabas command-level link routine. LNKOLM is the independent part of the Adabas command-level link routine.
CICSASMD	DISPGWA	Display program for Adabas global work area (GWA).
CICSASME	LNKENAB	Adabas PLT-enable program.
CICSASMT	LNKTRUE	Adabas task-related user exit.

Sample Resource Definitions

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

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

```
************************************************************************
* Sample DEFINE control statements for the DFHCSDUP utility.
* For Adabas V7.4 CICS command-level link routine components.
*
* These control statements can be used as input to the DFHCSDUP
* CICS CSD update utility to define the Adabas CICS command-level
* link routine components on a CICS/TS system.
************************************************************************
DEFINE PROGRAM(ADABAS) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s COMMAND LEVEL LINK ROUTINE)
LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(CICS) EXECUTIONSET(FULLAPI)
```

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

DEFINE PROGRAM(ADATEST) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s DISPLAY GWA PROGRAM - DISPGWA)
LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
ELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(CICS) EXECUTIONSET(FULLAPI)

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

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

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

These sample DEFINE statements are located in member DEFADAC in the Adabas Version 7.4 CICS command-level source library. They can be modified and used as input to the IBM DFHCSDUP utility to define the Adabas CICS command-level components. Consult the appropriate IBM CICS documentation for information on the DFHCSDUP utility.

Modifying Source Member Defaults (ADAGSET Macro)

The ADAGSET macro is used to create default settings for the command-level link components. This macro exists in each of the installation source members. The macro settings must be identical in every installation source member used.

To facilitate the assembly of the Adabas command-level link components, Software AG recommends that you modify the ADAGSET macro with site-specific keyword values and place the updated copy in a library that is available in the LIBDEF source search chain for the assembly job step.

If more than one version of the Adabas command-level link routine components is to be used in a given CICS system, modify a separate ADAGSET macro for each version and place these macros in unique sublibraries. Then reference the appropriate sublibrary in the LIBDEF search chain in the assembly JCS.

Note:
Beginning with Adabas version 6.2.3, the source members no longer contain any keyword values on the ADAGSET statement. Thus, the macro itself must be modified to provide site-specific defaults.

It is critical that the values for the following keywords agree for all components of the Adabas CICS command-level link routine: LOGID, SVCNO, LUINFO, LRINFO, LUSAVE, NUBS, ENTPT, TRUENAM, and ENABNAM.

Step 1 of the installation procedure identifies the source members that must be edited for standard and enhanced installation.

The ADAGSET parameter options with their default values (underlined) are described below:

AVB: Adabas VSAM Bridge Support

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

ENABNM: Entry Point Name for Program to Enable Adabas TRUE

Parameter	Description
ENABNM={ 'ADAENAB' | 'name' }	The entry point name for the program that is run to enable the Adabas TRUE during CICS PLTPI processing. The value must be a valid program name that matches the module name specified in the DFHPLT table at your site. This parameter is ignored if TRUE=NO is specified.

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

Parameter	Description
ENTPT={ 'ADABAS' | 'name' }	The name given to the Adabas CICS command-level link routine, which is the combination of LNKOLSC, LNKOLM, and the CICS entry and exit code. This name is used in EXEC CICS LINK commands to invoke Adabas services from CICS application programs. See also notes 1 and 2 in the installation procedure.

LADAFP: Length of Work Area for Adabas Fastpath Exit

Parameter	Description
`LADAFP={ 0 | nn}`	The length of the work area provided to the Adabas Fastpath exit. Values from 0 (the default) to 32767 may be specified. 0 indicates that Adabas Fastpath is not linked with the Adabas command-level link routine. A non-zero value requires that the parameter TRUE=YES is also set and the Adabas task-related user exit (TRUE) is used. Consult the Adabas Fastpath documentation for recommended values. **Note:** This parameter is not yet fully implemented. It is provided for future use by Adabas Fastpath.

LOGID: Default Logical Database ID

Parameter	Description
`LOGID= nnn`	The value of the default logical database ID. Valid ID numbers are 1-65535.

LRINFO: Length of Adabas Review Data Area

Parameter	Description
`LRINFO={ 0 | 256}`	The length (in bytes) of the Adabas Review data area to be used by the REVEXITB program. The default is zero (Adabas Review is not being used). The minimum (and recommended) value is 256, the size Adabas Review expects when the REVEXITB program is invoked. See the Adabas Review documentation for more information.

LUINFO: Length of User Data passed to Adabas UEXITA and UEXITB

Parameter	Description
`LUINFO={ 0 | length}`	Length of the user data to be passed from the CICS link routine to Adabas UEXITA and UEXITB. If LUINFO is not specified, the default is zero (no user save area is passed).

LUSAVE: Size of User Save Area for Adabas UEXITA and UEXITB

Parameter	Description
`LUSAVE={ 0 | size}`	Size of the user save area to be used by Adabas user exits UEXITA and UEXITB. If LUSAVE is specified, a value of 72 or higher must be specified. If LUSAVE is not specified, the default is zero (no user data is passed).

LXITAA: Length of Work Area provided to UEXITA

Parameter	Description
`LXITAA={ 0 | nn}`	Length of the work area provided to the UEXITA user exit program. Values from 0 (the default) to 32767 may be specified. 0 indicates that no UEXITA program is linked with the Adabas command-level link routine and no data is passed to UEXITA. **Note:** This parameter is not yet fully implemented. It is provided for future use by the CICS user exit A program linked with LNKOLM.

LXITBA: Length of Work Area for UEXITB

Parameter	Description
`LXITBA={ 0 | nn}`	Length of the work area provided to the UEXITB user exit program. Values from 0 (the default) to 32767 may be specified. 0 indicates that no UEXITB program is linked with the Adabas command-level link routine and no data is passed to UEXITB. **Note:** This parameter is not yet fully implemented. It is provided for future use by the CICS user exit A program linked with LNKOLM.

MRO: Multiple Region Option

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

NETOPT: Method Used to Create User ID

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

NTGPID: Natural Group ID

Parameter	Description
`NTGPID=4-byte-value`	This parameter is used to specify a 4-byte Natural group ID as required for unique Adabas user ID generation in the CICSplex environment with Natural Version 2.2.8 and above. The value is associated with all users who call the Adabas command-level link routine assembled with the specified value. There is no default value. If no value is specified, the Adabas internal user ID is built in the conventional manner. Any 4-byte alphanumeric value may be specified, but it must be unique for each Adabas command-level link routine running in a CICSplex, or z/OS image. If more than one NTGPID is required (for example, both test and production Natural 2.2.8), more than one Adabas command-level link routine with associated TRUE must be generated. If you run with the CICS multiple region option (MRO), you may use NTGPID to provide a 4-byte literal for the Adabas communication ID to be used by the Adabas SVC when multiple application regions call Adabas.

NUBS: Number of User Blocks Created By CICS Link Routine

Parameter	Description
`NUBS={ 50 | blocks }`	The number of user blocks (UBs) to be created by the CICS link routine. The number of blocks must be large enough to handle the maximum possible number of concurrent Adabas requests. **Note:** The Adabas 6.2 and above command-level link routine obtains storage for the user blocks (the UB pool) above the 16-megabyte line.

PARMTYP: Area for Adabas Parameter List

Parameter	Description
`PARMTYP={ ALL | COM | TWA }`	The area which is to contain the Adabas parameter list. TWA picks up the parameter list in the first six fullwords of the transaction work area (TWA). COM picks up the list in the COMMAREA, followed by the normal Adabas parameter list. The COMMAREA list must be at least 32 bytes long and begin with the label ADABAS52. PARMTYP=ALL (the default) uses both the COMMAREA and TWA to pass the Adabas parameters; in this case, the COMMAREA is checked first. PARMTYP=ALL or PARMTYP=COM must be used if the TRUE=YES option is specified.

PURGE: Purge Transaction

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

RMI: Resource Manager Interface

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

SAF: Adabas Security

Parameter	Description
SAF={ NO | YES }	The SAF paramter is to indicate whether or not the Adabas SAF Security (ADASAF) is used. The default SAF=NO must be used for z/VSE.

SAP: SAP Application Support

Parameter	Description
SAP={ NO | YES }	The SAP parameter is used to indicate whether or not Adabas support for the SAP application system is required. If SAP=YES is specified, the LNKOLSC program will detect a SAP initialization call and set the user ID for SAP applications from the constant provided on the initialization call, plus the field ACBADD2. For more information, refer to the supplementary information provided to customers using the SAP application system.

SVCNO: Adabas SVC number

Parameter	Description
SVCNO={ 0 | nnn}	The SVCNO parameter is used to specify the value of the Adabas SVC number.

TRUE: Adabas Task-Related User Exit

Parameter	Description
TRUE={ NO | YES }	The TRUE parameter is used to indicate whether or not the Adabas task-related user exit is to be used. If TRUE=YES is specified, LNKOLSC will use the Adabas task-related user exit LNKTRUE. If TRUE=YES is specified, the parameter settings PARMTYP={ALL | COM} and TRUENM='name' must also be specified.

TRUENM: Name of Adabas Task-Related User Exit

Parameter	Description
TRUENM= 'name'	The TRUENM parameter is used to specify the name of the Adabas task-related user exit. This parameter is required if TRUE=YES is specified. See also notes 1 and 2 in the installation procedure.

UBPLOC: User Block Pool Allocation

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

XWAIT: XWAIT Setting for CICS

Parameter	Description
`XWAIT={ NO | YES }`	Specifies whether a standard EXEC CICS WAIT EVENT (XWAIT=NO) or a CICS WAIT EVENTS EXTERNAL (XWAIT=YES) will be generated into the command-level link component by the assembler process in the LNKOLSC module: • NO (the default) must be used for CICS/VS 2.3 and below. • YES is valid for CICS TS 1.1 under z/VSE 2.4 and above. **Note:** All EXEC CICS commands are processed by the CICS preprocessor; the ADAGSET parameters cause the subsequent assembly step to skip some of the statements. XWAIT=YES is the recommended interface for CICS TS 1.1 under z/VSE 2.4 and above. Using the XWAIT=NO interface in these environments may result in poor CICS transaction performance or unpredictable transaction results in busy environments.

Installation Procedure

Step 1: Modify the ADAGSET Macro for the Source Member(s)

Modify the ADAGSET macro for the source members to be used.

See the section *Modifying Source Member Defaults (ADAGSET)* for details. Software AG recommends that you modify a common version of ADAGSET and place it in a library available in the SYSLIB concatenation when the Adabas command-level link components are assembled.

Note:
It is no longer necessary to modify the equates inside the LNKOLSC code. Instead, use the ADAGSET macro to set default values before assembly, thus making the process easier and more self-documenting.

For the Standard Installation (Without Enhanced Functions)

Modify the source member ADAGSET to set the following options:

- database ID (LOGID);
- Adabas SVC number (SVCNO);
- any additional options necessary for your site.

For the Enhanced Installation

Modify the source member ADAGSET to set the following options:

- database ID (LOGID);
- Adabas SVC number (SVCNO);
- the command-level link routine name (ENTPT);
- the task-related user exit name (TRUENM);
- any additional options necessary for your site.

Notes:

1. It is critically important that the ENTPT and TRUENM parameters coded in the LNKENAB, LNKTRUE, and LNKOLSC modules are identical. These module names are used to identify the global work area and task-related user exit (TRUE) storage provided for the CICS transaction using the link-specific link component.
2. If you are installing multiple instances of the command-level link routine, the ENTPT and TRUENM names must be unique, as there is a one-for-one relationship between a command-level link routine, its associated task-related user exit (TRUE), and the enabling program LNKENAB run at CICS startup.

Step 2: Modify the JCS Members

Use your editor to modify the JCS members necessary for your installation, and set the library names according to your installation’s specifications. The Adabas job library contains the following JCS members:

JCS Member	Installation Type	Required/Optional	This job preprocesses, assembles, and links ...
CICSASMC	standard and enhanced	required	enhanced LNKOLSC and LNKOLM to create ADAOLSC and ADAOLM; then, links these modules with the CICS prolog and epilog stubs to create the Adabas CICS command-level link component.
CICSASMD	enhanced	required	the Adabas global work area display program DISPGWA to create the ADATEST module.
CICSASME	enhanced	required	the LNKENAB PLTPI initialization program ADAENAB.
CICSASMT	enhanced	optional	the LNKTRUE source module to create the Adabas task-related user exit module ADATRUE.

Instructions for modifying the JCS are included in the JCS members themselves. These include but may not be limited to the following:

1. Correct the POWER control statements:

change ...	to ...
$*	/*
$&	/&
X $$	* $$

2. Replace the following symbols with site-specific values:

change ...	to ...
LLLLLLLL	Adabas library DLBL name
SSSSSSSS	Adabas sublibrary for cataloging
UUUUUUUU	user ID
EXT1	extent start value for work file 1
EXT2	extent start value for work file 2
EXT1L	extent length value for work file 1
EXT2L	extent length value for work file 2
VVVVVV	volume ID for work files

Step 3: Install the Adabas Command-Level Link Component

to install the Adabas command-level link component, for the JCS members CICSASMC, CICSASMD, CICSASME, and CICSASMT:

1. Include the proper EXEC PROC= statements to define the Adabas library.
2. Modify the LIBDEF statements for proper search order and for the proper sublibrary for cataloging the PHASEs.
3. Ensure that the library containing the PHASEs will be available to the CICS system at execution time.

   The JCS members use two disk work files for SYSPUNCH output from the preprocessor and assembler steps.
4. Check the DLBL and EXTENT information for these work files.
5. Use DFHCSDUP or the CEDA RDO entry panels to add the following definition to your CICS CSD file:

```
DEFINE PROGRAM(ADABAS) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s COMMAND LEVEL LINK ROUTINE)
LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(USER) EXECUTIONSET(FULLAPI)

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

The Adabas command-level link routine is now installed. This completes the standard installation. To install the enhanced functions, continue with step 4.

Step 4: Install the Adabas Task-Related User Exit

The Adabas CICS components reside in the Adabas base source library.

to install the Adabas Task-Related User Exit:

1. Execute CICSASME.

   This job preprocesses, assembles, and links the Adabas task-related user-exit-enabling program LNKENAB (ADAENAB).
2. Use DFHCSDUP or the CEDA RDO entry panels to add the following definition to your CICS CSD file:

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

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

3. After defining LNKENAB to CICS, add the following entry to your PLTPI table, DFHPLT:

   ```
   DFHPLT TYPE=ENTRY,PROGRAM=ADAENAB
   ```

   This entry should follow the first DFHPLT TYPE=DELIM statement to ensure that LNKENAB will be executed in either stage II or stage III of the CICS PLTPI process. This is necessary because the CICS EXEC interface environment must be present to support the writing of console messages using the EXEC CICS WRITE OPERATOR command employed by the LNKENAB module.

4. Code an appropriate PLTPI=*xx* parameter in the CICS start-up data. *xx* should match the suffix value given in the DFHPLT table.

   The Adabas command-level link enhanced functions are now installed. This completes the enhanced installation; however, you may optionally continue with Step 6, Installing the DISPGWA Program.

Step 5: Install the Adabas PLT-Enable Program (SMA Job Number I070)

The Adabas CICS components reside in the Adabas base source library ADA74*s*.SRCE.

to install the Adabas PLT-enable program:

1. Execute CICEASM.

   This job preprocesses, assembles, and links this module into a staging library, and then links it with CICS entry and exit code into a CICS RPL library.

2. Use DFHCSDUP or the CEDA RDO entry panels to add the following definition to your CICS CSD file:

   ```
   DEFINE PROGRAM(ADAENAB) GROUP(ADABAS)
   DESCRIPTION(ADABAS V74s PLTPI ENABLE ADATRUE PROGRAM)
   LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
   USE
   EXECKEY(CICS) EXECUTIONSET(FULLAPI)
   ```

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

3. After defining LNKENAB to CICS, add the following entry to your PLTPI table, DFHPLT:

   ```
   DFHPLT TYPE=ENTRY,PROGRAM=ADAENAB
   ```

   This entry should follow the first DFHPLT TYPE=DELIM statement to ensure that LNKENAB will be executed in either stage II or stage III of the CICS PLTPI process. This is necessary because the CICS EXEC interface environment must be present to support the writing of console messages using the EXEC CICS WRITE OPERATOR command employed by the LNKENAB module.

4. Code an appropriate PLTPI=*xx* parameter in the CICS start-up data. *xx* should match the suffix value given in the DFHPLT table.

Step 6: Install the DISPGWA Program (Optional)

to install the DISPGWA program:

1. Execute CICDASM.

   This job preprocesses, assembles, and links this module into a staging library, then links it with CICS entry and exit code into a CICS RPL library. The final link step creates the load module ADATEST.

2. Add a CICS transaction to execute the ADATEST program. RDO may be used to do this. Sample DEFINE statements for the ADATEST (DISPGWA) program and the transaction to execute it are:

```
DEFINE PROGRAM(ADATEST) GROUP(ADABAS)
DESCRIPTION(ADABAS V74s DISPLAY GWA PROGRAM - DISPGWA)
LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(CICS) EXECUTIONSET(FULLAPI)

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

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

   The Adabas command-level link routine, enhanced functions, and DISPGWA program are now installed.

Installing the CICS High-Performance Stub Routine

The Adabas high-performance stub routine extends the direct call interface (DCI) facility that is available with the Adabas CICS command-level link component to applications written in languages other than Software AG's Natural (for example, Assembler, COBOL, PL/I).

Note:
The stub routine must be used with the Adabas CICS command-level link component. The stub routine will not function properly with the Adabas CICS macro-level link component.

The DCI allows a CICS TS 1.1 application running under z/VSE 2.4 or above to call Adabas through the Adabas command-level link routine. The overhead incurred when the EXEC CICS LINK and EXEC CICS RETURN command set is used to transfer program control is thus avoided. Once the proper environment has been established with the initial call (IC) command from the high-performance stub or Natural 3.1 or above, the DCI permits a BALR interface to be used.

The high-performance stub routine is written in Assembler language. When linked with the application program, it serves as an interface between the application and the Adabas CICS command-level link component. The application program can then issue CALL statements to access the stub routine when executing an Adabas command.

A CICS TS 1.1 application under z/VSE 2.4 or above derives the following advantages from the high-performance stub:

- improved performance and throughput when issuing Adabas commands under CICS/ESA 3.2 or above due to the reduced use of CICS services related to the CICS LINK and RETURN program control mechanism.
- a call mechanism for Adabas requests under CICS/ESA 3.2 or above which is simpler than the methods normally employed to pass control with information from one program to another in the CICS environment.

Restrictions and Requirements

The following restrictions and requirements apply to the high-performance stub routine:

- CICS TS 1.1 under z/VSE 2.4 Required

  The Adabas high-performance stub routine is supported for CICS TS 1.1 under z/VSE 2.4 and above. Earlier versions of CICS are not supported.

  A CICS transaction work area (TWA) of at least 28 bytes must be provided to the application for the proper execution of the high-performance stub routine.

- CICS Command-Level Link Required

  The application program must be written using the CICS command-level interface and instructions, and may not issue any CICS macro level commands.

- Supported Programming Languages

  The application program may be written in ALC (Assembler language), COBOL, COBOL II, PL/I, or C. Installation verification programs (IVPs) are provided in ALC and COBOL on the distribution tape.

Additional requirements for specific programming languages are discussed later in the sections relating to each language.

Stub Components

Type	Member	Description
Source	ALCSIVP COBSIVP LNCSTUB	source for ALC install verification source for COBOL install verification source for high-performance stub
Job control	JCLALCI.X JCLCOBI.X JCLLNCS.X	sample JCL for ALC installation verification sample JCL for COBOL installation verification sample JCL LNCSTUB (high-performance stub). The final step of the job catalogs the LNCSTUB.OBJ module so it is ready to be included when the application program is linked. See the JCLALCI or JCLCOBI sample jobs.

Installation Overview

Use the following procedure to install the Adabas CICS high-performance stub routine:

- Edit, preprocess, assemble and link the LNCSTUB module.
- (Optional) Modify, preprocess, compile or assemble, link, and execute the desired installation verification program (IVP).
- Modify, preprocess, compile or assemble, link, and execute the application programs.

Step 1: Install the LNCSTUB Module

The Adabas CICS high-performance stub routine is an Assembler language module provided in source form on the distribution tape in member LNCSTUB.

Step 1 has the following substeps:

- Edit the ADAGSET macro.
- Change the LNCNAME field value, if necessary.
- Modify member JCLLNCS.X.
- Preprocess, assemble, and link the LNCSTUB module.
- Place the LNCSTUB load module in a library that is available to your application programs when they are linked.

Edit the ADAGSET Macro

Note:
For information about editing the ADAGSET macro, refer to the section *Modifying Source Member Defaults (ADAGSET Macro)*.

Edit the LNCSTUB module to provide parameters on the ADAGSET macro located at the beginning of the source deck. The values provided to the ADAGSET macro should match those provided on the Adabas CICS command-level link routine LNKOLSC.

The values given for the ADAGSET parameters are primarily for documentation purposes within the LNCSTUB module, but may be used at a later time in the stub routine at the discretion of Software AG.

Change the LNCNAME Field Value

If your Adabas CICS command-level link component program has been linked with a name other than ADABAS, change the constant value in the field LNCNAME to match the name used (see the ADAGSET option ENTPT). The value in this field is used in the priming EXEC CICS LINK command issued by LNCSTUB.

Modify Member JCLLNCS.X

Member JCLLNCS.X is used to preprocess, assemble, and catalog the LNCSTUB module.

To modify this JCL to meet your site requirements

1. Modify the POWER job information, CLASS, DEST, and LDEST values:
2. Throughout the member, make the following changes:

change ...	to ...
$*	/*
$&	/&
X $$	* $$

3. Provide any site-specific names for the Adabas JCS PROC or other required PROCs.
4. Replace the following symbols with site-specific values:

change ...	to ...
SSSSSSSS	Adabas sublibrary for cataloging
UUUUUUUU	user ID
EXT1	extent start value for work file 1
EXT2	extent start value for work file 2
EXT1L	extent length value for work file 1
EXT2L	extent length value for work file 2
VVVVVV	volume ID for work files

5. Software AG recommends that you use the high-level Assembler to assemble the LNCSTUB module.

   If you must use the old VSE Assembler, supply the following statement on the assembly step to provide the assembly date:

```
// OPTION SYSPARM='yymmdd'
—where
yy is the 2-digit year
mm is the 2-digit month
dd is the 2-digit day
```

6. Check the access and catalog statements on the final LIBR step that catalogs the LNCSTUB.OBJ module.

Preprocess, Assemble, and Link the LNCSTUB Module

Because of the possible use of the 31-bit instructions, high-level Assembler (ASMA90) should be used to assemble the LNCSTUB module after CICS preprocessing.

to preprocess, assemble, and link the LNCSTUB Module

1. Provide library definitions for the Adabas and CICS libraries using PROCs or DLBL statements.
2. Construct LIBDEF statements for the search order and for the library where the LNCSTUB.OBJ module is to be cataloged.

Make the LNCSTUB Available to Application Programs

The LNCSTUB module has an entry name of ADABAS, which can be used by the application program as the object of a CALL statement to pass control to LNCSTUB with a list of parameters. The language-specific calling conventions for LNCSTUB are discussed later in this section.

The LNCSTUB load module must be available to the link step of the application program that is to use the DCI facility.

Note:
In the same step, the CICS load library should be available; otherwise, the external references to the CICS stub modules will not be resolved.

Place the LNCSTUB load module in a library available to your application language assembler or compiler so that it will be included when the application programs are linked.

Step 2: (Optional) Install and Execute an IVP

Two installation verification programs (IVPs) are provided in source form: one for Assembler language, and one for COBOL. These programs are samples for implementing the Adabas high-performance stub routine in your applications. They also provide a way of verifying the proper installation of the LNCSTUB module.

Step 2 has the following substeps:

- Modify the Assembler (ALCSIVP) or COBOL (COBSIVP) source decks to provide the proper Adabas database ID and file number on your site’s database for the Software AG-provided PERSONNEL file.
- Modify the JCL provided to preprocess and compile (assemble) the desired IVP.
- Preprocess, compile or assemble, and link the IVP using the sample JCL provided as a guide.
- Add PPT and PCT entries to your CICS system to execute the IVPs.
- Execute the IVPs to verify the LNCSTUB module (ALCSIVP and COBSIVP).

Install and Execute the Assembler IVP: ALCSIVP

The source member ALCSIVP is provided to demonstrate and verify the use of the Adabas DCI using the LNCSTUB module. This program issues a series of Adabas commands using the conventional CICS LINK/RETURN mechanism, produces a partial screen of output data, then reexecutes the same call sequence using the Adabas DCI and the LNCSTUB subprogram.

to modify source member ALCSIVP:

1. Edit the database ID and file number fields DBID (line 321) and DBFNR (line 322) to be sure they match the values needed to access the PERSONNEL file on the database you intend to use.
2. Check the fields FBUFF, SBUFF and VBUFF for values consistent with your PERSONNEL file's FDT and data content.
3. Check the name used in the EXEC CICS LINK statement (line 242) to be sure it matches the name of your Adabas CICS command-level link component program.

to modify the JCLALCI.X sample job stream:

1. Member JCLALCI.X is used to preprocess, assemble, and link the Assembler IVP. Place the ALCIVP.PHASE in a library that will be available to your CICS partition.
2. Modify JCLALCI.X to meet your site requirements.

   Modify the POWER job information, CLASS, DEST, and LDEST values:

   Throughout the member, make the following changes:

change ...	to ...
$*	/*
$&	/&
X $$	* $$

   Provide any site-specific names for the Adabas JCS PROC or other required PROCs.

   Replace the following symbols with site-specific values:

change ...	to ...
SSSSSSSS	Adabas sublibrary for cataloging
UUUUUUUU	user ID
EXT1	extent start value for work file 1
EXT2	extent start value for work file 2
EXT1L	extent length value for work file 1
EXT2L	extent length value for work file 2
VVVVVV	volume ID for work files

3. Software AG recommends that you use the high-level Assembler to assemble the LNCSTUB module.

   If it is necessary to use the old VSE Assembler, supply the following statement on the assembly step to provide the assembly date:

```
// OPTION SYSPARM='yymmdd'
—where
yy is the 2-digit year
mm is the 2-digit month
dd is the 2-digit day
```

to preprocess, assemble, and link ALCSIVP:

1. Using the modified sample JCLALCI.X member, preprocess, assemble, and link ALCSIVP.

to add PPT and PCT entries:

1. Add the following PPT and PCT entries to your CICS system, or use the RDO facility to add the STB1 transaction to run the ALCSIVP program:

```
DFHPPT TYPE=ENTRY,PROGRAM=ALCSIVP
DFHPCT TYPE=ENTRY,PROGRAM=ALCSIVP,TRANSID=STB1, X
TPURGE=YES,SPURGE=NO,TWASIZE=28
```

to execute ALCSIVP:

1. Run the STB1 transaction to execute ALCSIVP. Executing ALCSIVP verifies the LNCSTUB module.

Install and Execute the COBOL IVP: COBSIVP

Member COBSIVP illustrates the use of the Adabas DCI with a COBOL program. COBIVP produces a screen showing output lines produced by a series of Adabas calls executed by the CICS LINK/RETURN facility, followed by the reexecution of these Adabas commands using the DCI.

to modify source member COBSIVP:

1. Edit the fields WORK-DBID and WORK-FNR to place the desired database ID and file number in the VALUE clauses to access the PERSONNEL file on your site’s database.

2. Ensure that the value in the field LINK-NAME matches the name used in your Adabas CICS command-level link component program.

3. Ensure that the values (literals in the PROCEDURE DIVISION) in the following fields are consistent with the requirements of the PERSONNEL file FDT and data content you are using:

```
ADABAS-FORMAT-BUFFER,
ADABAS-SEARCH-BUFFER, and
ADABAS-VALUE-BUFFER
```

to modify the JCLCOBI.X sample job stream:

1. Member JCLCOBI.X is used to preprocess, compile, and link the COBSIVP installation verification program.

   Modify JCLCOBI.X to meet your site requirements.

   Modify the POWER job information, CLASS, DEST, and LDEST values:

   Throughout the member, make the following changes:

change ...	to ...
$*	/*
$&	/&
X $$	* $$

   Provide any site-specific names for the Adabas JCS PROC or other required PROCs.

   Replace the following symbols with site-specific values:

change ...	to ...
cuu	channel and unit address for SYSPCH
lib	Adabas library DLBL name
sublib	Adabas sublibrary for cataloging
UUUUUUUU	user ID
rtrk	relative track number for the extent
ntrks	number of tracks in the extent
volid	volume ID for work files

   Replace the names in lowercase with the site-specific library and dataset names.

2. Member JCLCOBI.X is used to preprocess, compile, and link the COBSIVP installation verification program.

to preprocess, compile, and link COBSIVP:

1. Use the modified JCLCOBI.X job to preprocess, compile, and link the COBSIVP program. Assemble ADASTWA into a library available to COBOL programs when they are linked. Include the ADASTWA load module in the link of COBSIVP.

   COBSIVP uses the ADASTWA subroutine when it issues Adabas calls through the standard CICS LINK/RETURN mechanism. ADASTWA is supplied in source form in the Adabas source library.

   The LNCSTUB subroutine does not use ADASTWA because it places the passed Adabas parameters in the TWA. Thus, the ADASTWA routine is not required when linking COBOL applications that utilize the Adabas DCI through the LNCSTUB module.

   Even though the LNCSTUB routine does not require the ADASTWA subroutine, it is included in the COBSIVP program to illustrate the usual way in which a COBOL application places the Adabas call parameters into the CICS TWA.

2. Link the COBSIVP program with the LNCSTUB object module and the ADASTWA object module. Make the LNCSTUB load module available to the linkage editor to be included with the COBSIVP load module.

   Note:
   The CICS stub modules are also resolved in the link step.

to add PCT and PPT entries:

1. Add the following PPT and PCT entries to your CICS system, or use the RDO facility to add the STB2 transaction to run the COBSIVP program:

```
DFHPPT TYPE=ENTRY,PROGRAM=COBSIVP
DFHPCT TYPE=ENTRY,PROGRAM=COBSIVP,TRANSID=STB2, X
TPURGE=YES,SPURGE=NO,TWASIZE=28
```

to execute COBSIVP:

1. Run the STB2 transaction to execute COBSIVP. Executing COBSIVP verifies the LNCSTUB module.

Step 3: Link and Execute the Application Program

Once the IVP programs have been successfully executed, the Adabas DCI is ready to be used with real application programs. In step 3, the application program interface (API) is coded to utilize the LNCSTUB subprogram.

Step 3 has the following substeps:

- Modify the application programs that will utilize the Adabas CICS high-performance stub routine in accordance with the guidelines described in the following section.
- Preprocess, compile or assemble, and link the application programs to include the LNCSTUB module.
- Execute the application programs using the Adabas CICS high-performance stub.

Guidelines for Modifying the Application Program

The LNCSTUB load module must be linked with your application program. The application program invokes the DCI interface using a standard batch-like call mechanism. The LNCSTUB module makes any additional CICS requests required to pass data to the Adabas CICS command-level link component.

- Programming Languages Supported by LNCSTUB

  The LNCSTUB program functions with application programs written in Assembler language, VS/COBOL, COBOL II, PL/I, and C.

- Transaction Work Area Required

  A transaction that uses the Adabas DCI or the Adabas CICS command-level link component must provide a transaction work area (TWA) at least 28 bytes long. Failure to provide an adequate TWA will result in an abend U636 (abnormal termination of the task).

- Reentrant Requirement

  The application program may or may not be reentrant. The LNCSTUB module has been written to be reentrant, but using linkage editor parameters to mark the LNCSTUB load module as reentrant is not recommended.

- CICS Requests Issued by LNCSTUB

  The LNCSTUB module issues the following command-level CICS requests whenever it is invoked:

  ```
  EXEC CICS ADDRESS TWA
  EXEC CICS ASSIGN TWALENG
  EXEC CICS ADDRESS EIB
  ```

- DCI Entry Point Address

  An EXEC CICS LINK command is issued by LNCSTUB at least once to acquire the DCI entry point from the Adabas CICS command-level link component program. This address is then used for BALR access on all subsequent Adabas calls for a transaction. Thus, the calling application program must provide a fullword (4-byte) field to hold the DCI entry point address obtained by LNCSTUB. This 4-byte field is the first parameter passed to the LNCSTUB module by the call mechanism. The remaining parameters comprise the standard Adabas parameter list needed to execute an Adabas request.

- DCI Parameter List

  The Adabas DCI parameter list expected by the LNCSTUB program is composed of a pointer to the DCI entry point in the Adabas CICS command-level link component followed by the six pointers to the Adabas control block and buffers: format, record, search, value, and ISN.

  For information on coding the standard Adabas control block and buffers, refer to the *Adabas Command Reference*.

  The parameter list offsets are summarized in the table below:

Offset	Pointer to the ...
0	DCI entry point in the Adabas command-level link component
4	Adabas control block
8	Adabas format buffer
12	Adabas record buffer
16	Adabas search buffer
20	Adabas value buffer
24	Adabas ISN buffer

All of the parameters except the first (the DCI entry point) are built and maintained by the application program in accordance with the requirements of an Adabas call.

The DCI entry point parameter should be set to binary zeros at the beginning of a task, and should not be modified by the application program thereafter. Software AG strongly recommends that the fields comprising the parameter list be placed in CICS storage (WORKING-STORAGE for COBOL and the DFHEISTG user storage area for Assembler) to maintain pseudo-reentrability.

The following is a sample parameter list for an assembler language program:

```
DFHEISTG DSECT
.
PARMLIST DS 0F
DS A(DCIPTR)
DS A(ADACB)
DS A(ADAFB)
DS A(ADARB)
DS A(ADASB)
DS A(ADAVB)
DS A(ADAIB)
.
DCIPTR DS F
ADACB DS CL80
ADAFB DS CL50
ADARB DS CL250
ADASB DS CL50
ADAVB DS CL50
ADAIB DS CL200
.
DFHEIENT CODEREG=(R12),EIBREG=(R10),DATAREG=(R13)
.
LA R1,PARMLIST
L R15,=V(ADABAS)
BALR R14,R15
.
END
```

Note:
The DFHEIENT macro in the Assembler example uses a DATAREG parameter of register 13. This is a strict requirement of the LNCSTUB program. When the LNCSTUB program is invoked, register 13 should point to the standard CICS save area (DFHEISA) and register 1 should point to the parameter list. The best way to ensure this standard is to code the Assembler application with a DFHEIENT macro like the one in the example.

The following is a sample parameter list for a COBOL language program:

```
WORKING-STORAGE SECTION.
.
01 STUB-DCI-PTR PIC S9(8) COMP VALUE ZERO.
01 ADACB PIC X(80).
01 ADAFB PIC X(50).
01 ADARB PIC X(250).
01 ADASB PIC X(50).
01 ADAVB PIC X(50).
01 ADAIB PIC X(200).
.
PROCEDURE DIVISION.
.
CALL 'ADABAS' USING STUB-DCI-PTR,
ADACB,
ADAFB,
ADARB,
ADASB,
ADAVB,
ADAIB.
.
EXEC CICS RETURN END-EXEC.
.
GOBACK.
```

- Restrictions on Application Program Coding

  In all other respects, the application program should be coded like a standard CICS command-level routine. As long as the DCI parameter list is correct when LNCSTUB is called, there are no restrictions on the CICS commands that an application can issue.

- Standard Batch Call Mechanism Used

  As shown in the Assembler and COBOL language program parameter list examples above, the call to "ADABAS" (the LNCSTUB entry point) is accomplished like a batch application. Likewise, calls for the other supported languages should be coded with their standard batch call mechanisms.

Link the Application Programs to Include the LNCSTUB Module

To properly link the LNCSTUB module with application programs, link the application program to include the LNCSTUB module and the CICS stub modules. The method for doing this varies with the programming language used for the application:

- Assembler language programs should include the DFHEAI and DFHEAI0 CICS modules;
- COBOL applications should include DFHECI and DFHEAI0.

To avoid a double reference to the DFHEAI0 module, code the linkage editor REPLACE DFHEAI0 control statement at the beginning of the SYSLIN data deck.

for linking Assembler language programs:

1. For an Assembler program, the SYSLIN input is similar to:

```
INCLUDE DFHEAI
```

For an Assembler program, the input to the linkage editor is

```
PHASE pgmname,*
MODE AMODE(ANY),RMODE(ANY)
INCLUDE DFHEAI
INCLUDE LNCSTUB
INCLUDE DFHEAI0
ENTRY pgmname

—where pgmname is the name of your application program.
```

The Assembler object deck looks like

```
REPLACE DFHEAI0
INCLUDE SYSLIB(LNCSTUB)
INCLUDE SYSLIB(DFHEAI0)
NAME ALCSIVP(R)
```

When examining the cross-reference from the linkage editor, the symbol ADABAS must have the same starting location as the LNCSTUB module in the link map.

for linking COBOL language programs:

1. For a COBOL program, the input to the linkage editor is

```
PHASE pgmname,*
MODE AMODE(ANY),RMODE(ANY)
INCLUDE LNCSTUB
INCLUDE DFHECI
ENTRY pgmname

—where pgmname is the name of your application program.
```

The COBOL object input is similar to:

```
INCLUDE SYSLIB(LNCSTUB)
INCLUDE SYSLIB(DFHEAI0)
NAME COBSIVP(R)
```

When examining the cross-reference from the linkage editor, the symbol ADABAS must have the same starting location as the LNCSTUB module in the link map.

for linking PL/I and C language programs:

1. Refer to the IBM manual *CICS/ESA System Definition Guide* for information about linking PL/I and C applications under CICS.

Performance Using LNCSTUB

To obtain the best performance from applications using the Adabas direct call interface (DCI), examine how the DCI interface functions at the logical level.

A CICS application using the standard LINK/RETURN mechanism to access the Adabas link routines invokes the CICS program control service for every Adabas request made to the link routine. The LNCSTUB module permits a BALR interface to be used. A BALR interface can substantially reduce the

CICS overhead required to pass control from the application program to the Adabas CICS command-level link component.

The LNCSTUB module accomplishes this by using the standard EXEC CICS LINK/RETURN mechanism to make an Initial Call (IC) to the Adabas CICS command-level link routine. The link routine recognizes this call, and returns the entry point address of the DCI subroutine to LNCSTUB. LNCSTUB must then save this address in a location that can be assured of existence throughout the duration of the invoking task. This is why the calling program must provide the 4-byte field to hold the DCI entry point address. After the DCI address has been obtained, and for as long as LNCSTUB receives this address as the first parameter passed to it on subsequent Adabas calls, LNCSTUB utilizes the BALR interface to pass control to the Adabas CICS command-level link component program.

As a consequence of this logic, the more Adabas requests made between ICs, the more efficient the application in terms of passing data to and from Adabas under CICS. In fact, pseudo-conversational applications that issue one Adabas call each time a task is invoked should not be coded to use the DCI because there will be an IC request for each Adabas command issued by the calling program.

An additional performance improvement can be realized by taking advantage of the fact that the Adabas CICS command-level link component program must be defined as resident in CICS. This fact should allow the DCI entry point to be stored across CICS tasks, making it possible for different programs to call the LNCSTUB module with a valid DCI entry point. The IC at each program startup is thus avoided. When this procedure is used, however, any change to the CICS environment that invalidates the entry point address (such as a NEWCOPY) will lead to unpredictable and possibly disastrous results.

It is imperative that at least one IC be made to the Adabas CICS command-level link component program using CICS services. This call is used to trigger the acquisition of shared storage for the Adabas user block (UB) and (in the case of migration aids) an array of register save areas. If no IC request is made, Adabas calls will not execute due to a lack of working storage, and to the fact that critical control blocks used by the link routines and the Adabas SVC are not built.

Installing Adabas with Com-plete

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

For Com-plete Versions 4.5 and above, the link routine for Com-plete initialization (module/phase ADALCO) is provided in the Adabas distribution library. ADALCO is loaded during Com-plete initialization to service Adabas calls. The Adabas library containing ADALCO should be placed in the z/VSE LIBDEF search chain.

Com-plete branches to the module/phase ADALCO according to the AMODE setting established in the link of ADALCO. The Adabas Version 7.4 ADALCO routine uses the ESA BASSM and BSM instructions to

- call the Com-plete wait service in the proper AMODE; and
- return to the caller of ADALCO in the caller’s AMODE.

Installing Adabas with Shadow

The Adabas link routine specific to Shadow, ADALNS, is provided in source form along with the job SHADASM to assemble it. SHADASM must be customized to select the Shadow macro (source) library containing the macros SAVED, $WAIT, RELOCD, RETURND, TCBD.

Selecting Options for ADALNS

Customizing the source member ADALNS means selecting the following options:

Option	Default	Specify . . .
SVCNR	0	Adabas SVC number
LOGID	1	Default logical database ID (range 1-255).
NUBS	50	Number of UBs (user blocks) to be created by ADALNS. This must be high enough to handle the maximum possible number of concurrent Adabas requests.
PLINTWA	Y (yes)	N if the Adabas parameter list is passed in register 1 instead of at offset 0 in the Shadow TWA.
LNUINFO	0	Length for the user data to be passed from the ADALNS link routine to the Adabas user exit 4.

Shadow Table Entry for ADALNS

The user must specify the following entry in the Shadow PCT table:

```
PCT PROG=ADABAS,DISP=INITL,LANG=BAL,SAVE=YES,PURGE=YES
```

It is important that Adabas be made resident. Under Shadow, the ADABAS parameter is normally passed in the first 24 bytes of the TWA.

The user exit called from ADALNS gains control before the Adabas call (UEXITB), and can be used to modify the eight-byte UBUID field. This allows users who process the command log to have a unique terminal, since the command log presently contains only a four-byte field. This field does not contain a unique ID. The user exit could then be used to make the first four bytes unique. The user exit must create a unique user exit for each user. For Shadow, the UBUID field normally contains the constant SHAD in the high-order four bytes, followed by the value from ITRMTYPE.

Installing Adabas with Batch / TSO

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

However, it is important to note that user programs linked with ADAUSER also load ADARUN. ADARUN, in turn, loads other modules. To start a user program linked with ADAUSER, the following modules must be available in the LIBDEF search chain:

```
ADAIOR
ADAIOS
ADALNK
ADAMLF
ADAOPD
ADAPRF
ADARUN
```