

Installing Adabas with TP Monitors

This chapter provides information needed to install Adabas in batch mode and with its teleprocessing (TP) monitors. It covers the following topics:

- Preparing Adabas Link Routines for IBM Platforms
 - Installing Adabas with IMS TM under Adabas 8
 - General Considerations for Installing Adabas with CICS
 - Installing Adabas with CICS under Adabas 8
 - Installing the CICS High-Performance Stub Routine for Adabas 8
 - Installing Adabas with Com-plete under Adabas 8
 - General Considerations for Installing Adabas with Batch/TSO
 - Installing Adabas with Batch/TSO under Adabas 8
 - Establishing Adabas SVC Routing by Adabas Database ID
 - Modifying Source Member Defaults (LGBLSET Macro) in Version 8
-

Preparing Adabas Link Routines for IBM Platforms

This section describes the preparation of Adabas link routines for TP monitors for IBM platforms. The source modules for Adabas 8 link routines are not provided in the Adabas 8 base source library. The Adabas 8 link routines can only be tailored via zap or using a link globals table.

- Addressing Mode Assembly Directives in Adabas Link Routines
- UES-Enabled Link Routines

Addressing Mode Assembly Directives in Adabas Link Routines

All Adabas 8 link routines include AMODE and RMODE assembly directives. These assembly directives allow the linkage editor to produce warning messages when conflicting AMODE or RMODE linkage-editor control statements are encountered in the link JCL, 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.

Re-linking Adabas 8 Link Routines

When re-linking the Adabas 8 link routines with certain AMODE and RMODE combinations, a warning message may be generated by the linkage editor. This may be safely ignored as long as it pertains to a conflict of AMODE or RMODE in the ESD record of one or more of the load modules that comprise the link routine, and as long as the resulting module has the proper AMODE and RMODE attributes for execution

with the intended calling application programs.

Care must be taken to ensure that AMODE (24) applications will operate properly when invoking the link routine with the attributes chosen when it is re-linked. This is particularly important if the RMODE (ANY) attribute is associated with a link routine that will be loaded dynamically but invoked by a program that is AMODE (24). In this case, the link routine should be re-linked AMODE (31), RMODE (24) to avoid addressing exception ABENDs because the AMODE (24) application cannot correctly invoke the link routine if it resides above the 16-megabyte line.

The Adabas 8 link routines all run AMODE (31) after initialization, but they will return to the caller in the caller's AMODE.

Note:

Under CICS, the version 8 links run AMODE (31), but the Dataloc RDO parameter governs the AMODE and RMODE of the running CICS transaction.

The batch/TSO non-reentrant link routine, ADALNK, has been assembled and linked with AMODE (31), RMODE (24), and that is the recommended configuration to support AMODE (24) or RMODE (24) application programs. It may be re-linked AMODE (31), RMODE (ANY) if desired, but this should only be done if it is certain that all calling programs are AMODE (31).

The ADALNKR batch TSO reentrant link routine has been assembled and link-edited with AMODE (31), RMODE (ANY). If it is loaded by an application that is AMODE (24), it should be relinked AMODE (31), RMODE (24).

The z/OS Com-plete module ADALCO has been assembled and linked AMODE (31), RMODE (ANY). The Com-plete TP monitor ensures proper AMODE switching between AMODE (24) or RMODE (24) programs that invoke ADALCO through the Com-plete Adabas interface routine, TLOPADAB.

All of the version 8 CICS link routine modules - ADACICS, ADACICT, and ADACIC0 - have been assembled and link-edited AMODE (31), RMODE (ANY). CICS manages the loading of programs and their invocation depending on the DATALOC values associated with their program and transaction definitions.

The Adabas IMS interface link routine ADALNI has been assembled and link-edited AMODE (31), RMODE (ANY). This is the preferred configuration for modern IMS applications, but if there are still AMODE (24) IMS applications executing at your installation, ADALNI may be re-linked AMODE (31), RMODE (24).

ADAUSER AMODE/RMODE Considerations

Software AG recommends that all batch applications invoke Adabas calls through the ADAUSER module. This module is normally link-edited with the application program and it then loads the appropriate link routine as well as ADARUN and ADAIOR/ADAIOS. The source member has the AMODE and RMODE directives coded as AMODE 31, RMODE ANY. This is the most flexible configuration for assembling and linking ADAUSER with the widest variety of application programs. However, if ADAUSER is dynamically loaded, either the RMODE assembler directive should be changed to RMODE 24 before re-assembling it or the ADAUSER module should be re-linked AMODE (31), RMODE (24) to ensure that AMODE 24 application programs may invoke it properly below the 16-megabyte line.

UES-Enabled Link Routines

The source code for the Adabas 8 batch and TSO link routines is not included with Adabas 8. These modules are delivered with LNKUES along with the ASC2EBC and EBC2ASC translation tables. Please run with these UES components for this version of Adabas 8. The link routine will detect if a Version 8 target database is not UES-enabled, and will provide an Adabas response code 228 (ADARSP228) if the call is from a client requiring UES translation.

The Adabas 8 Com-plete link routine determines whether UES support is required from the settings in the LCOGBL module that you modify and assemble when installing Adabas with Com-plete. For complete information, read *Installing Adabas with Com-plete*.

This section covers the following topics:

- Default or Customized Translation Tables
- Calling LNKUES and LNKUES7
- Adabas 8 Jobs for z/OS Universal Encoding Support
- Disabling UES Support for Adabas 8 Routines

Default or Customized Translation Tables

By default, the load modules for all Adabas 8 link routines have been linked with LNKUES and the default translation tables.

LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller.

The two standard translation tables are:

- ASC2EBC: ASCII to EBCDIC translation; and
- EBC2ASC: EBCDIC to ASCII translation.

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

You may use the load modules with the default translation tables linked in, or you may prepare your own customized translation tables, re-assemble the tables, and link them with the LNKUES or LNKUES7 module that is delivered.

Notes:

1. 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.
2. The LNKUES module is functionally reentrant; however, it is not linked that way in the Adabas load library.
3. When linking the LNKUES load module and the translation tables, the linkage editor may produce warning messages concerning the reentrant or reusability status of the linked module. These warning messages can be ignored.
4. If relinking an Adabas 8 link routine for UES support, the LNKUES module must be included. This will ensure that your new Adabas 8 applications have support for Adabas 8 direct calls and control blocks.

Calling LNKUES and LNKUES7

LNKUES is called only on Adabas link routine 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. In Adabas 8 requests, LNKUES receives control before UEXIT1. In Adabas 8 replies, LNKUES receives control after UEXIT2.

Adabas 8 Jobs for z/OS Universal Encoding Support

The following lists the sample jobs provided to manage universal encoding support in Adabas link routines in z/OS environments:

Sample Job	Description
LNKGCICS	Assembles and links the CICS globals table with LNKUES and the default translation tables ASC2EBC and EBC2ASC.
LNKLCO8	Links the Com-plete link globals table with LNKUES and the default translation tables ASC2EBC and EBC2ASC.
LNKLN18	Links the IMS link routine with the LNIGBL link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.
LNKLNK8	Links the batch link routine with the LNKGBLS link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.
LNKLNKR8	Links the reentrant batch link routine with the LNKRGBL link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.

Before you can use any of these jobs, they should be edited to prepare the JOB card, update the load library names, and make other changes as necessary for your environment. Refer to the comments in the jobs themselves for more information.

Disabling UES Support for Adabas 8 Routines

This section describes how to disable UES support in the Adabas 8 IMS TM, Com-plete, and batch/TSO link routines, if for some reason you feel it is necessary.

To disable UES support in link routines:

1. Edit the link globals table for the associated link routine. Set the UES parameter setting to NO.
2. Assemble the link globals table 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 with the newly assembled link globals table and do not include any of the UES components (that is, LNKUES, ASC2EBC, or EBC2ASC).

For more information about the specific link routines, read *Installing Adabas with IMS TM under Adabas 8*, *Installing Adabas with Com-plete under Adabas 8*, and *Installing Adabas with Batch / TSO under Adabas 8*.

Installing Adabas with IMS TM under Adabas 8

This section describes installation of the Adabas link routine for the IMS TM TP monitor with Adabas 8.

IMS requires an Adabas link routine if it is to communicate with Adabas databases. The Adabas Version 8 executable default link routine is delivered in member ADALNI of the AII vrs .LOAD library (where vrs is the number of the latest Adabas *version* delivered on the tape). If you want to modify this link routine, use member ADALNI8 to do so. ADALNI8 must be linked with a link globals module you prepare and with any link routine exits you require to create the final ADALNI load module that is loaded by the IMS message processing program (MPP) regions when an application calls them. Members ADALNI and ADALNI8 are provided with some default settings.

This section covers the following topics:

- IMS TM Link Routines for Adabas 8
- Obtaining the Adabas User ID
- Obtaining the SAF ID
- Installation Procedure under Adabas 8

IMS TM Link Routines for Adabas 8

These are Adabas 8 link routines for IMS TM:

- ADALNI is the executable default module for message processing programs (MPPs). If you require no changes to the defaults provided in the link routine, use this module.
- Use ADALNI8 as the base module for message processing programs (MPPs). If you need to tailor ADALNI for your installation, use ADALNI8 to generate an updated ADALNI.
- ADALNK is the batch Adabas link routine for batch message processing (BMP) programs, batch-oriented BMP programs, and batch processing programs (DLIBATCH).

ADALNI and ADALNK use the CSECT name and ENTRY directive ADABAS by default.

The Adabas Version 8 ADALNI and ADALNK are UES-enabled as distributed. See the section *Enabling Universal Encoding Support (UES) for Your Adabas Nucleus* for more information.

This section describes using ADALNI and ADALNI8 only. For information on using ADALNK, read *General Considerations for Installing Adabas with Batch/TSO*.

Obtaining the Adabas User ID

The Adabas user ID is obtained at execution time by the ADALNI load module from the LTERM field (first eight bytes) of the IOPCB. The user ID is stored in the Adabas user block field UBUID and will be used for the last eight bytes of the Adabas communication ID.

Obtaining the SAF ID

The SAF ID is supported for use by Adabas SAF Security (ADASAF) if an external security package such as IBM's RACF or CA's ACF2 is present. The SAF ID is obtained at execution time by the ADALNI load module from the user ID field (bytes 33-40) in the IOPCB. To get a valid SAF user ID, SAF sign-on must be active in your IMS installation and the user must have performed an IMS /SIGN command to log onto an IMS terminal.

Installation Procedure under Adabas 8

To modify the default settings and prepare the Adabas 8 link routine for IMS:

1. Copy the sample member LNIGBL provided in the Adabas 8 AIIvrs.SRCE library to any appropriate user source library where they can be modified. These modules contain LGBLSET parameters that are used to create default settings for link components. A complete description of LGBLSET parameters can be found in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*.
2. Modify the LNIGBL member in the user source library.

Note:

The OPSYS parameter must be set to ZOS.

3. Modify and run sample job ASMGMLS as described at the top of the job. ASMGMLS can be found in the Adabas 8 ADAvrs.JOBS library. When fully modified, the SET statement in the job should reference the LNIGBL member you prepared in the previous step and the NAME link edit control statement should reference the name specified by the GBLNAME parameter in the LNIGBL member.

Once modified, submit the ASMGMLS job to assemble and link-edit the link globals module.

A new link globals module (with the name specified by the GBLNAME parameter in LNIGBL) will be generated in the user load library identified in the ASMGMLS job.

4. Copy sample job LNKLNI8 to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the ADALNI8 base module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from this job to an appropriate user load library. LNKLNI8 can be found in the Adabas 8 AIIvrs.SRCE library.

The module resulting from this job is ADALNI.

5. Place the ADALNI module in a load library available for IMS MPP regions.

The Adabas 8 link routine is prepared.

General Considerations for Installing Adabas with CICS

The macro-level link routine ADALNC is no longer supported for all levels of CICS running under z/OS. These environments must run a current version of Adabas and use the supplied command-level link component.

The Adabas command-level link routine supports the CICS transaction server (CTS) environment.

Notes:

1. The OPID option for the USERID field is not supported under CICS/TS 1.1 and above; therefore, it is not provided with the command-level link routine.
2. The CICS components from Adabas 7.4 or later are required when running with an Adabas 8 SVC.

The following sections describe specific points of Adabas/CICS installation and operation from the CICS perspective:

- Adabas Bridge for VSAM Considerations
- CICS MRO Environment Requirements
- Using CICS Storage Protection
- Sample Resource Definitions
- Requirement for CICS Command Resource Security

Adabas Bridge for VSAM Considerations

If you are running Adabas Bridge for VSAM 4.2 or 5.1 under CICS, you must run CICS 3.3 or above and the Adabas Version 7.1 or above command-level link routine.

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 MRO parameter to "YES" and use the default for the NETOPT parameter. In an Adabas 8 installation, these parameters are supplied via the LGBLSET macro (read *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*).

You can use the LGBLSET NTGPID parameter 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 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.

This exit is link user exit 1 (LUEXIT1). 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.

Using CICS Storage Protection

The storage protection mechanism (STGPROT) was introduced under CICS/ESA 3.3. Storage protection permits resources to access either CICS or user storage by using the storage protection keys.

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

Transaction isolation is an extension of the storage protection mechanism. It further protects CICS resources 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 CICS 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.

In Adabas 8 installations, the CICS link routine always uses a task-related user exit, module ADACICT, so storage protection is supported by default.

Sample Resource Definitions

Under CICS/TS 1.1 and above for z/OS, 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.

Modify and use the sample DEFINE statements located in member DEFADAC 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. The DEFADAC member can be found in the Adabas 8 CICS command-level source library (ACIvrn.SRCE).

Requirement for CICS Command Resource Security

The Adabas CICS link routines require a command security level of "UPDATE" for the EXITPROGRAM CICS command resource identifier. This allows the Adabas CICS application stub to issue the EXEC CICS EXTRACT EXIT command without raising the NOTAUTH response from CICS and the security software. The Adabas CICS application stub needs to issue the EXEC CICS EXTRACT EXIT to determine that the given Adabas task-related user exit (TRUE) is installed and enabled, and to locate the CICS global work area (GWA) associated with the given TRUE so that various data structures are made available to the Adabas CICS application stub programs.

Installing Adabas with CICS under Adabas 8

A CICS application that uses Adabas services requires an *Adabas CICS execution unit* to function.

In Adabas versions prior to 8.2, the Adabas CICS execution unit was comprised of:

- the Adabas CICS stub, ADACICS
- the stub module's direct call interface ADADCI
- the Adabas task-related user exit (TRUE), ADACICT
- the globals table, named CICSGBL by default.

The stub module needs to know the name of the Adabas TRUE it is to invoke. In addition, the Adabas TRUE needs to know the name of the globals table so that it can obtain run-time information, such as the locations of callable exits and the settings of various operating parameters (such as the length of user information).

Effective with Adabas 8.2 and later versions, the Adabas CICS execution unit is comprised of:

- the Adabas CICS stub, ADACICS
- the stub module's direct call interface, ADADCI
- an Adabas CICS names module, ACINAMES
- one or more Adabas task-related user exits (TRUEs), ADACICT
- a globals table associated with the stub module and the TRUE.

The names module (ACINAMES) is linked with the stub (ADACICS) to provide the name of the associated TRUE and the globals table for a given CICS application. In addition, an Adabas CICS installation options table (ACIOPT) is required and used by the Adabas CICS installation program, ADACIC0, to load the Adabas globals tables required by the Adabas CICS execution units that will be installed and activated in the CICS region.

This section covers the following topics:

- The Adabas CICS Application Stub (ADACICS)
- The Adabas CICS Names Module (ACINAMES)
- The Adabas CICS Installation Options Table (ACIOPT)
- The MACINS Macro
- The MACIOPT Macro
- Adabas Task-Related User Exits (TRUEs)
- Supplied Modules
- Installation Procedure

The Adabas CICS Application Stub (ADACICS)

The Adabas CICS application stub is invoked via EXEC CICS LINK or via the direct-call interface from a CICS application program that intends to use Adabas database services. The stub consists of the ADACICS module, the ADADCI module, the CICS modules DFHEAI and DFHEAI0, and the ACINAMES module. The resultant load module may be given any name that is specified in the link globals ENTPT keyword for the Adabas execution unit. The new module name is most easily created with the linkage editor.

The Adabas CICS Names Module (ACINAMES)

The Adabas CICS names module (ACINAMES) is a small stub containing the name of the TRUE to be invoked from this stub and the name of the link globals table associated with the Adabas CICS execution unit. The link globals table also contains the names of the stub and the TRUE, but linking it with the stub has the following performance disadvantages:

- The stub is functionally reentrant and the link globals table in CICS is modifiable during execution
- Linking the globals table with the stub would also cause duplicate copies of the link globals table to be kept in CICS storage at the same time, wasting space and possibly leading to problems if the copy loaded by ADACIC0 differs from the copy linked with the Adabas stub

Using the ACINAMES module allows you to relink the Adabas CICS stub with any supported load module name and gives that stub the ability to invoke the Adabas CICS TRUE with the name provided in the ACINAMES module. The TRUE may also be relinked with any given valid load module name. This permits the CICS region to execute different Adabas stubs and TRUES built out of the same load modules but tailored as required for different CICS applications. No changes are needed in the CICS application programs themselves.

The Adabas CICS names module is built using the MACINS macro. The ACINAMES module may be given any load module name, but the generated CSECT name (ordinarily generated by the MACINS macro assembly job, ASMCINS) within the load module must be ACINAMES.

The Adabas CICS Installation Options Table (ACIOPT)

An additional component, an Adabas CICS installation options table (ACIOPT) is required and used by the Adabas CICS installation program, ADACIC0, to load the Adabas globals tables required by the Adabas CICS execution units that will be installed and activated in the CICS region.

The Adabas CICS installation options table is built using the MACIOPT macro.

The MACINS Macro

Use the MACINS macro to build the Adabas CICS names module, ACINAMES. The ACINAMES module may be given any load module name, but the generated CSECT name (ordinarily generated by the MACINS macro job) within the load module must be ACINAMES. In addition, the ACINAMES module should be included when the Adabas CICS stub is relinked.

The MACINS macro is provided in the Adabas CICS z/OS source library. A sample ACINAMES source member is provided in the ACIvrs source library on z/OS systems.

The syntax of the MACINS macro is shown below:

```
MACINS GTNAME = link-globals-table-name
      TRUENAME = true-module-name
```

All MACINS parameters are required and are described in the following table:

Parameter	Description	Default
GTNAME	<p>Specifies the name of the link globals table associated with this Adabas CICS stub.</p> <p>This parameter is required.</p> <p>The name specified by the GTNAMES parameter must be the name of a module that has been defined to CICS. It must also match the name of a link globals table specified in the Adabas CICS Installation Options Table (ACIOPT).</p>	There is no default.
TRUENAME	<p>Specifies the name of the Adabas CICS task-related user exit (TRUE) to be invoked by this Adabas CICS stub.</p> <p>This parameter is required.</p> <p>The name specified by the TRUENAME parameter must be the name specified in the TRUENM parameter of the link globals table specified in the corresponding GTNAME parameter</p>	There is no default.

Example

In the following example, an ACINAMES module is prepared for an Adabas CICS stub named ADABAS that will use an ADABAS CICS TRUE named ADATRUE and a link globals table named CICSGBL. The source member to create the ACINAMES module might look like this:

```
*          Sample "ACINAMES" for Adabas 8.2 multiple-TRUE support.
          MACINS TRUENAME=ADATRUE,                               X
          GTNAME=CICSGBL
```

The MACIOPT Macro

Use the MACIOPT macro to build the Adabas CICS installation options table which may either be linked with ADACIC0 or, if named ACIOPT (the default), is defined to CICS and loaded by ADACIC0 when the Adabas CICS installation process is started.

The MACIOPT macro is located in the ACIvrs source library on z/OS systems. A sample ACIOPT source member is provided in the ACIvrs source library on z/OS systems.

The syntax of the MACINS macro is shown below:

```
MACIOPT ENTRY = GLOBAL,GEN = { CSECT | DSECT }
                    ,CNAME = { ACIOPT | module-name }
                    ,MSGDEST = { CONSOLE | TDQ | BOTH }
                    ,IMQNAME = queue-name
                    ,MNTRUE = { 8 | number }

                    GROUP ,GTNAME = link-globals-table-name

FINAL
```

An ENTRY statement is required on every invocation of the MACIOPT macro. It designates the ENTRY type, which in turn, determines which additional parameters are valid for the given entry. The three types of ENTRY statement and their associated parameters are described in the rest of this document.

- The ENTRY=GLOBAL Statement
- The ENTRY=GROUP Statement
- The ENTRY=FINAL Statement
- Example

The ENTRY=GLOBAL Statement

The ENTRY=GLOBAL statement is always the first entry for the ACIOPT source member. Only one ENTRY=GLOBAL statement should be specified per source member and it should precede all other MACIOPT statements.

The ENTRY=GLOBAL statement specifies global parameters to be used by the CICS installation program. The parameters associated with ENTRY=GLOBAL are described in the table below:

Parameter	Description	Default
GEN	Indicates whether the ACIOPT CSECT or a mapping DSECT of the ACIOPT module should be generated. Valid values are CSECT or DSECT.	CSECT
CNAME	Identifies the load module name to be generated when link-editing a module directly with ADACIC0. Any module name can be specified, but ACIOPT is the recommended name (and the default). An ENTRY ACIOPT statement is generated in the CSECT of the load module to ensure that the V-CON in ADACIC0 will be satisfied when a module with a different name is linked. We recommend that you use the default load module name of ACIOPT, defining ACIOPT to CICS and allowing ADACIC0 to load the ACIOPT module when the program is executed to install the Adabas CICS components.	ACIOPT

Parameter	Description	Default
IMSGDEST	<p>Identifies the destination type for the installation progress and error messages produced by ADACIC0: console, transient data queue, or both.</p> <p>IMSGDEST=CONSOLE is the default and causes all installation messages to be written to the console with EXEC CICS WRITE OPERATOR commands. This is how messages for previous Adabas CICS components produced installation messages.</p> <p>IMSGDEST=TDQ causes ADACIC0 to determine if a named CICS transient data queue is available and, if so, to write installation progress and error messages to that queue. If IMSGDEST=TDQ is specified, the IMQNAME parameter must also be specified to provide the name of the CICS transient data queue for the messages. If the named transient data queue is not enabled and open, messages will be written to the console. No error message is written to indicate that the transient data queue could not be used. If the CICS transient data queue is open and enabled, message ADAK001 is written to the console to indicate that all further messages will be written to the CICS transient data queue. If, during ADACIC0 processing, the transient data queue becomes unavailable, subsequent messages will be written to the console.</p> <p>IMSGDEST=BOTH causes installation progress messages to be written both to the console and to a named CICS transient data queue.</p>	CONSOLE
IMQNAME	<p>Specifies the 4-character name of the CICS transient data queue where installation progress and error messages should be written. If IMQNAME is specified then the IMSGDEST parameter must be set to TDQ or BOTH.</p> <p>The named transient data queue must be defined to CICS as either an extra-partition queue or as an indirect queue which references an extra-partition data queue. The simplest way to set up such a data queue is to make it indirect and refer to the CICS-supplied extra-partition data queue CSSL.</p> <p>The queue may be defined using the CICS RDO facility (using the CEDA transaction) or using the DFHDCT macro. For more information, consult the appropriate IBM CICS documentation.</p> <p>Installation messages written to a CICS transient data queue are variable length records with no printer control character in the first byte of the record. The records will not exceed 132 bytes in length.</p>	There is no default.

Parameter	Description	Default
MNTRUE	<p>Specifies a maximum value for the number of Adabas CICS execution units (and thus globals tables) to be installed for this CICS or CICSplex.</p> <p>If this number is exceeded, a warning MNOTE and condition code of 4 is produced by the assembler.</p> <p>This parameter is provided as an option to place an upper limit on the number of Adabas CICS execution units that may be installed. You might find this necessary to limit the storage and resource constraints multiple Adabas CICS execution units might place on your system. Although the setting for MNTRUE may be quite high, the storage, resources and Adabas CICS components must be available to be installed.</p>	8

The ENTRY=GROUP Statement

ENTRY=GROUP statements define the names of the Adabas globals tables that should be loaded and used to install the Adabas CICS execution units. More than one ENTRY=GROUP statement can be specified in the ACIOPT source member; all ENTRY=GROUP statements must be specified after the ENTRY=GLOBAL statement and before the ENTRY=FINAL statement.

Only one parameter can be specified for ENTRY=GROUP:

Parameter	Description	Default
GTNAME	<p>Specifies the name of the link globals table to be loaded and used to install an Adabas CICS execution unit.</p> <p>This parameter is required. Only one GTNAME parameter can be specified on each ENTRY=GROUP statement.</p>	There is no default.

The ENTRY=FINAL Statement

The ENTRY=FINAL statement must be the last MACIOPT statement in the source member. It causes the actual ACIOPT CSECT statements to be generated. Only one ENTRY=FINAL statement may be specified in the source member.

There are no parameters for the ENTRY=FINAL statement

Example

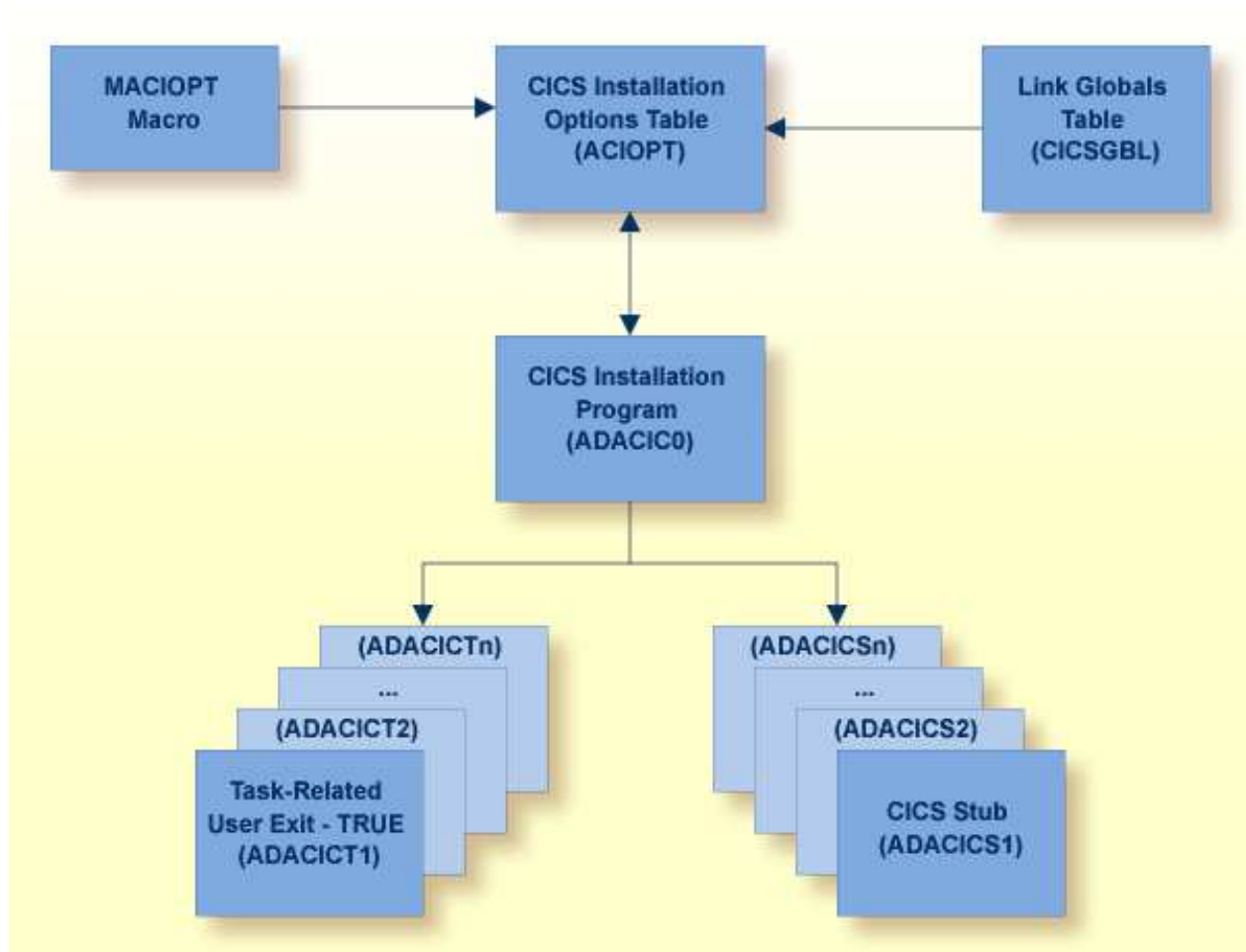
If assembled and link-edited, the following source member will produce the load module ACIOPT and will install two Adabas CICS execution units. One will load a globals table named LNKCI02 and the other will load a globals table named CICSGBL. Installation messages will be written to the CICS transient data queue named ACIQ, if that queue is available.

```
MACIOPT ENTRY=GLOBAL,IMSGDEST=TDQ,IMQNAME=ACIQ,MNTRUE=2
MACIOPT ENTRY=GROUP,GTNAME=LNKCI02
MACIOPT ENTRY=GROUP,GTNAME=CICSGBL
MACIOPT ENTRY=FINAL
```

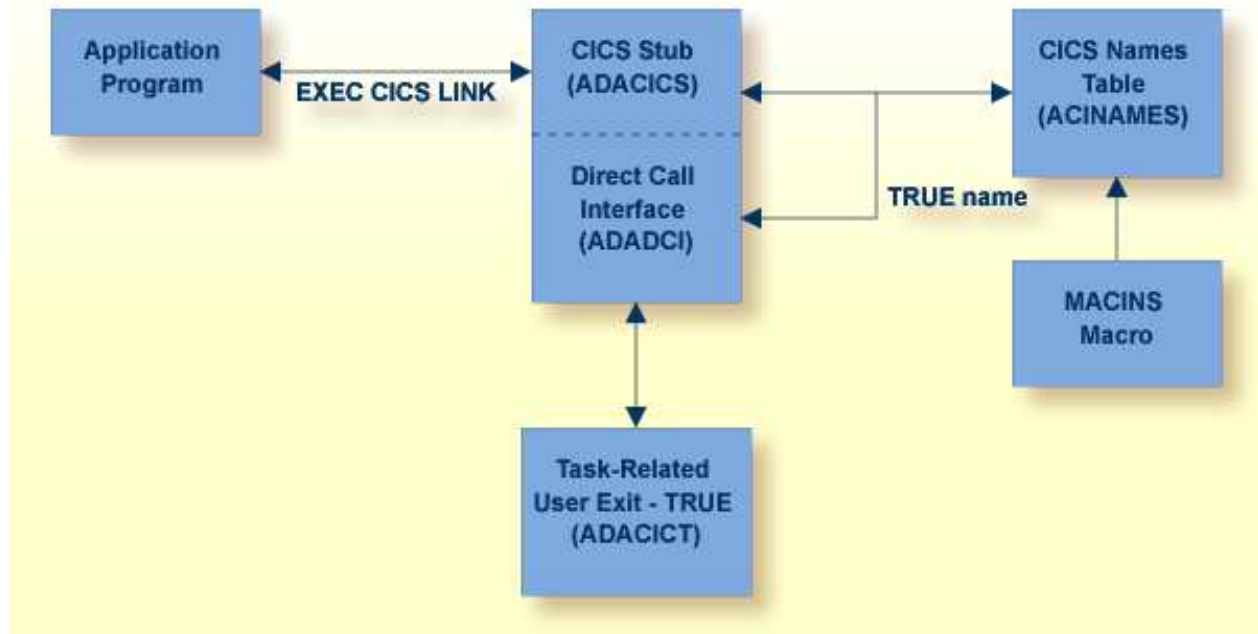
Adabas Task-Related User Exits (TRUEs)

Adabas 8.2 introduces support for the installation of multiple CICS task-related user exits (TRUEs) and Adabas application stubs from a single execution of the ADACIC0 installation program. Multiple TRUEs allow your site to tailor different Adabas CICS execution options in the same CICS region with a centralized installation procedure and software.

The following diagram depicts the processing flow of the installation of multiple Adabas CICS TRUE and application stub support.



The following diagram depicts the processing flow of the execution of this multiple Adabas CICS TRUE and application stub support.



Supplied Modules

The following table lists the modules supplied in your Adabas installation to support the installation of Adabas with CICS under Adabas 8.

Module	Description
ADACIC0	CICS installation program
ADACICS	CICS command-level module.
ADACICT	CICS task-related user exit (TRUE) module.

Installation Procedure

To install the Adabas 8 CICS link routine components, complete the following steps:

- Step 1. Copy the Load Modules
- Step 2. Prepare the Adabas CICS Installation Options Table
- Step 3. Prepare the Adabas CICS Task-Related User Exits (TRUEs) -- ADACICT
- Step 4. Prepare the Adabas CICS Names Module -- ACINAMES
- Step 5. Prepare the Adabas CICS Application Stub -- ADACICS
- Step 6. Prepare the CICS Link Globals Table -- CICSGBL
- Step 7. Assemble the CICS Link Globals Table -- ASMGBL
- Step 8. Link the Assembled CICS Link Globals Table -- LNKGCICS
- Step 9. Modify CICS Installation Values -- DEFADAC
- Step 10. Update the CICS CSD File
- Step 11. Modify, Assemble and Link the CICS PLTPI Table for ADACIC0
- Step 12. Start the CICS

Step 1. Copy the Load Modules

Copy the Adabas 8 CICS load modules from the Adabas distribution library to a load library that will be in the CICS DFHRPL concatenation (see sample member CPYCICSM in the Adabas 8 ADA v rn.JOBS library).

Step 2. Prepare the Adabas CICS Installation Options Table

An Adabas CICS installation options table (ACIOPT) is required to identify all the Adabas globals tables that will be needed for the proper execution of each Adabas CICS execution unit in the CICS region or CICSplex. The installation program (ADACIC0) run in Step 12 will obtain information of a global nature from the table such as the destination for writing of installation messages. It will also scan the table and load each Adabas globals table named in the ACIOPT module. In turn, each loaded globals table serves as the basis for installing each Adabas CICS execution unit.

The Adabas CICS installation options table is built by coding a series of MACIOPT macros into a source member, then assembling and linking that source member into a library that will be available during CICS execution. The load module may be linked:

- With the ADACIC0 installation program, or
- As a standalone module named "ACIOPT", which is then defined as a program of the same name to CICS.

For best performance, Software AG recommends linking a standalone ACIOPT module, defining it to CICS as program ACIOPT. This will allow ADACIC0 to load ACIOPT during the installation process.

To prepare the Adabas CICS installation options table, complete the following steps:

1. Code a source member, preferably called ACIOPT that contains MACIOPT macro statements to be loaded by the ADACIC0 program at execution time. The MACIOPT macro statements define each globals table that will be needed by each Adabas CICS execution unit.

The ACIOPT source member will consist of one MACIOPT ENTRY=GLOBAL entry, multiple MACIOPT ENTRY=GROUP entries and one MACIOPT ENTRY=FINAL entry.

- The MACIOPT ENTRY=GLOBAL specification must be first specification in the source member; only one MACIOPT ENTRY=GLOBAL specification can be made per ACIOPT generation.
- The MACIOPT ENTRY=FINAL specification must be the last entry for the ACIOPT generation; only one MACIOPT ENTRY=FINAL specification can be made per ACIOPT generation.
- Multiple MACIOPT ENTRY=GROUP entries may be specified, but they must follow the MACIOPT ENTRY=GLOBAL specification and precede the MACIOPT ENTRY=FINAL specification in the source member.

The MACIOPT macro is located in the ACI v rs source library on z/OS systems. For complete information on the MACIOPT macro, read *The MACIOPT Macro*, elsewhere in this section. A sample ACIOPT source member is provided in the ACI v rs source library on z/OS systems.

2. Assemble and link the ACIOPT source module either as the standalone module named "ACIOPT" or with any load module name linked with ADACIC0. If linked as a standalone module it must be named "ACIOPT" (see sample job ASMCOPT located in the ACIvrs source library) and it must be defined as a program to CICS.

The ACIOPT module may be defined to CICS using the CEDA/RDO facility or the DFHCSDUP utility. Sample DFHCSDUP statements are provided in the DEFADAC member in the ACIvrs source library on z/OS systems.

Step 3. Prepare the Adabas CICS Task-Related User Exits (TRUEs) -- ADACICT

An Adabas task-related user exit (TRUE) is created by relinking the Adabas ADACICT module with a NAME statement, providing the desired TRUE name. One or more Adabas TRUEs can be created.

Note:

The Adabas TRUE name is specified later in the TRUENM parameter in the link globals table (set Step 6) and in the TRUENAME parameter when the ACINAMES module (see Step 4) is prepared.

▶ To prepare the Adabas CICS TRUE, complete the following steps:

1. Relink the ADACICT module with a NAME or PHASE statement giving a new name for each Adabas TRUE.
2. Define each named Adabas TRUE as a program to CICS.

A sample job, LNKATRU, is provided in the ACIvrs source library. This sample links the Adabas TRUE with a load module named ADATRUE so that it can be installed and referenced in CICS.

Step 4. Prepare the Adabas CICS Names Module -- ACINAMES

The ACINAMES module is a small stub containing the name of the TRUE to be invoked from this stub and the name of the link globals table associated with the Adabas execution unit. After the ACINAMES source member is coded, it should be provided as input to the assembler and either punched by the assembler to a text library or directly link-edited as a load module. The subsequent text deck or load module would then be made available to the linkage editor when the Adabas CICS stub is relinked to change its name or to update the ACINAMES module it uses.

▶ To prepare the ACINAMES module, complete the following step:

- Code the source for the ACINAMES module using the MACINS macro. For complete information, read *The MACINS Macro*.

The MACINS macro is provided in the Adabas CICS z/OS source library.

Sample job, ASMCINS, which is provided in the ACIvrs source library, assemble the ACINAMES module and links it with the Adabas CICS application stub (see Step 5), and names the stub "ADABAS".

Example

For example, the source member to create the ACINAMES module might look like this:

```
*      Sample "ACINAMES" for Adabas 8.2 multiple-TRUE support.
      MACINS TRUENAME=ADATRUE,
      GTNAME=CICSGBL
```

X

This ACINAMES module uses an ADABAS CICS TRUE named ADATRUE and a link globals table named CICSGBL.

Step 5. Prepare the Adabas CICS Application Stub -- ADACICS

The Adabas application stub is invoked via EXEC CICS LINK or via the direct-call interface from a CICS application program that intends to use Adabas database services. The application stub consists of the ADACICS module, the ADADCI module, the CICS modules DFHEAI and DFHEAI0 and the ACINAMES module. The resultant load module may be given any name that is specified in the link globals ENTPT keyword for the Adabas execution unit. The new module name is most easily created with the linkage editor.

To prepare the CICS application stub (ADACICS), complete the following step:

- Relink the Adabas CICS application stub module, ADACICS, replacing ACINAMES in the module with the name of the ACINAMES module created in the previous step (Step 4).

Sample job, ASMCINS, which is provided in the ACIvrs source library, assemble the ACINAMES module and links it with the Adabas CICS application stub (see Step 4), and names the stub "ADABAS".

Example

For example, the link-edit control statements to create the Adabas module as the Adabas CICS stub might be:

```
//LKED.SYSIN DD *
MODE AMODE(31),RMODE(ANY)
REPLACE ACINAMES
INCLUDE ADALIB(ADACICS)
INCLUDE USERLIB(ACINAMES)
ENTRY ADACICS
NAME ADABAS(R)
/*
```

In this example, the prepared ACINAMES module is used for an Adabas CICS stub named ADABAS.

Step 6. Prepare the CICS Link Globals Table -- CICSGBL

Link globals tables must be prepared to match the Adabas CICS execution units defined in the ACIOPT module. These are built by editing or creating source members that use the LGBLSET macro and its keywords.

Modify the sample CICSGBL member found in the Adabas 8 ACIvrn.SRCE library. This member contains sample default installation (LGBLSET) parameter settings. For more information about what to modify in this member, read *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*.

Notes:

1. Adabas 8.2 no longer supports the ADACIRQ module or the reading of an input CICS transient data queue to obtain the name of the link globals table during installation. This was necessary to permit the installation of multiple Adabas CICS execution units from the same installation program.
2. The LGBLSET macro is included in the Adabas CICS link routine source library only for the Adabas 8.1.4 patch in which this feature was introduced. In all other releases, the LGBLSET macro is located in the Adabas source library.

 **To prepare the link globals table, complete the following steps:**

1. Code the link globals table using the LGBLSET macro as described in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*.

The OPSYS parameter must be set to ZOS.

Be sure to code the ENTPT and TRUENM parameters on each LGBLSET macro so they match the intended Adabas CICS stub name and Adabas CICS TRUE name to be used in a given Adabas CICS execution unit. The Adabas CICS installation program attempts to load each globals table in turn and uses the loaded table to provide the data required to install and activate the components of the execution unit.

2. Save the modified CICSGBL member with a unique name in an appropriate user source library.

Step 7. Assemble the CICS Link Globals Table -- ASMGBLS

Modify and run sample job ASMGBLS as described at the top of the job. ASMGBLS can be found in the Adabas 8 ADAvrs.JOBS library. When fully modified, the SET statement in the job should reference the CICSGBL member you prepared in Step 6 and the NAME link edit control statement should reference the name specified by the GBLNAME parameter in the CICSGBL member.

Step 8. Link the Assembled CICS Link Globals Table -- LKNGCICS

Review and run the LKNGCICS member in the ACIvm.SRCE library to link the newly assembled globals table from the previous step with any user or Software AG product exits. (For information about specific Software AG product exits, read the installation documentation for the product.) The LKNGCICS member provides specific instructions. Be sure to link the globals table into a load library that will be made available to CICS in the DFHRPL library concatenation. Note that any user or Software AG link routine exits should be link-edited with this load module.

Step 9. Modify CICS Installation Values -- DEFADAC

Modify the DEFADAC member to provide the correct name of the link routine globals default table created in Step 6. The default module name is CICSGBL. Tailor this member for any other CICS installation values as required.

Step 10. Update the CICS CSD File

Run the IBM DFHCSDUP utility to update the CICS CSD file for the desired CICS using the modified DEFADAC member as input.

Step 11. Modify, Assemble and Link the CICS PLTPI Table for ADACIC0

Modify the CICS PLTPI table to add an entry for the CICS installation program ADACIC0. The ADACIC0 installation program will start the TRUES once CICS is started. Use member ADAPLTXX from the Adabas 8 ACIvrn.SRCE library as a sample for enabling and starting a legacy Adabas TRUE and the new Version 8 TRUE in the second phase of the PLT.

Once the PLTPI table is modified, assemble and link the modified PLTPI table into a library that will be available to the desired CICS region.

Step 12. Start the CICS

Start the CICS and note any messages relating to the installation of the Adabas TRUE modules that appear on the console. When CICS starts, it will call ADACIC0 (because it is in the PLTPI table), which will install the Adabas CICS TRUES.

Installing the CICS High-Performance Stub Routine for Adabas 8

This section describes installation of the CICS high-performance stub routine with Adabas 8. The modules and installation described here are provided so your existing Adabas 8 applications can continue to function as usual.

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/VSE macro-level link component. The LNCSTUB module delivered in the Adabas Version 8 library will also function properly with Adabas Version 7.4 CICS link routines.

The DCI enables a CICS/TS application 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.

An application at CICS/TS 1.1 level or above derives the following advantages from the high-performance stub:

- improved performance and throughput when issuing Adabas commands under CICS/TS 1.1 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/TS 1.1 or above which is simpler than the methods normally employed to pass control with information from one program to another in the CICS environment.

This section covers the following topics:

- Restrictions and Requirements
- Stub Components
- Installation Overview
- Performance Using LNCSTUB
- Modifying Source Member Defaults (ADAGSET Macro)

Restrictions and Requirements

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

1. CICS/TS 1.1 or above required

The Adabas high-performance stub routine is supported under CICS/TS 1.1 or above.

A CICS transaction work area (TWA) of at least 24 bytes or a CICS COMMAREA of at least 32 bytes must be provided to the application for the proper execution of the high-performance stub routine. The Adabas 8 LNCSTUB module and the Adabas 8 installation verification programs now use the CICS COMMAREA instead of the CICS TWA to pass data between the IVP programs, LNCSTUB, and the CICS link routines. The use of the CICS COMMAREA has the following advantages over the use of the CICS TWA:

- The size of the COMMAREA can be set on a call-by-call basis by the application program, while the TWA size is set when the CICS transaction is defined.
- Applications using the CICS COMMAREA may run in stages II or III of CICS PLTPI processing. The CICS TWA is not available during PLTPI processing.
- The dynamic sizing of the CICS COMMAREA is better suited to the unbounded format of the Adabas 8 ACBX direct call, ACBX control block, and Adabas Buffer Descriptions (ABDs). For more information on the Adabas Version 8 direct call interface and the data structures it uses, read the *Adabas Command Reference Guide*

2. 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.

3. Supported Programming Languages

The application program may be written in ALC (Assembler language), VS/COBOL, COBOL II, COBOL/LE, PL/I, or C. Installation verification programs (IVPs) are provided in ALC and COBOL in the ACIvrs.SRCE library

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

Stub Components

Type	Member	Description
Source	ADAGSET ALCSIVP COBSIVP LNCSTUB	macro required for assembling LNCSTUB and ALCSIVP source for the ALC install verification source for the COBOL install verification source for the high-performance stub
Job control	JCLALCI JCLCOBI JCLLNCS	sample JCL for ALC install verification sample JCL for COBOL install verification sample JCL for LNCSTUB (high-performance stub)

Installation Overview

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

1. Edit, preprocess, assemble and link the LNCSTUB module.
2. Define the application programs, optional IVPs and CICS link components to CICS using RDO or the DFHCSDUP utility.
3. (Optional) Modify, preprocess, compile or assemble, link, and execute the desired installation verification program (IVP).
4. Modify, preprocess, compile or assemble, link, and execute the application programs.

This procedure is described in the following steps:

- Step 1: Install the LNCSTUB Module
- Step 2: (Optional) Install and Execute an IVP
- Step 3: Link and Execute the Application Program

Step 1: Install the LNCSTUB Module

The Adabas CICS high-performance stub routine is an Assembler language source module, provided in member LNCSTUB in the ACI_{vr}s.SRCE library.

Step 1 has the following substeps:

- Edit the ADAGSET Macro
- (Optional) Set the LNCSTUB Entry-Point Alias
- Modify Member JCLLNCS
- Preprocess, Assemble, and Link the LNCSTUB Module
- Make the LNCSTUB Available to Application Programs

Edit the ADAGSET Macro

Note:

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

Edit the ADAGSET macro in a library that will be available in the SYSLIB concatenation when LNCSTUB is assembled.

Both the LNCSTUB and the ALCSIVP IVP modules now take values from the following ADAGSET keywords:

- LOGID, which identifies the database ID
- PARMTYP, which determines whether the TWA or COMMAREA is used by the LNCSTUB and the ALCSIVP programs to pass data
- ENTPT, which specifies the name of the CICS link routine or CICS stub to be invoked by the LNCSTUB and ALCSIVP programs. If your Adabas CICS command-level link component program has been linked with a name other than ADACICS, change the value of the ENTPT keyword in the ADAGSET macro. The value in this field is used in the priming EXEC CICS LINK command issued by LNCSTUB.
- TRUENM, which specifies the name of the Adabas TRUE to use

(Optional) Set the LNCSTUB Entry-Point Alias

The Adabas 8 LNCSTUB module provides an assembler GBLC variable (&STBNAME) that sets an entry-point alias that can be used by calling programs. Modify the SETC statement near the top of the LNCSTUB source member to set an alias if desired. The application program can then either issue its call using "LNCSTUB" or the entry-point alias coded in this SETC statement.

Modify Member JCLLNCS

Member JCLLNCS (in the ADA vrs .JOBS library) is used to preprocess, assemble, and link the LNCSTUB module. To modify this JCL to meet your site requirements, change the JOB card in the member and the symbolic values as indicated in the following table:

Value	Description
&SUFFIX	Suffix value used for the CICS translator. The default value is “1\$”.
&ASMBLR	Assembler program used to assemble the LNCSTUB source (ASMA90).
&M	Member name to be processed; code LNCSTUB or ALCSIVP.
&STUBLIB	A load library to contain the LNCSTUB load module. This library should be available to application programs when they are linked.
&INDEX	High-level qualifier for the CICS macro library used in the SYSLIB DD statement for the assembler.
&INDEX2	High-level qualifier for the CICS load library to use for the translator STEPLIB DD statement, and for the SYSLIB in the link step.
&ADACOML	Adabas command-level source library containing the ADACB, ADAGDEF, ADAGSET, and LNCDS copy code and macros.
&ADASRCE	Adabas source library used for additional copy code or macro expansion.
&STBSRCE	Source library containing the distributed Adabas CICS high-performance stub LNCSTUB.
&MAC1	Primary system macro library, usually SYS1.MACLIB.
&OUTC	Output class for messages, SYSPRINT, SYSOUT.
®	Step region size.
&NCAL	Value for the linkage editor NCAL parameter. The recommended value is NCAL.
&LSIZE	Primary and secondary table sizes used by the linkage editor.
&WORK	DASD device type to use for temporary and utility data sets.

Preprocess, Assemble, and Link the LNCSTUB Module

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

Note:

The LNCSTUB module can be linked reentrant or reusable. If it is linked reentrant, it is automatically reusable; if it is linked reusable, it is not automatically reentrant.

In addition to the CICS macro library, the Adabas CICS command-level source library and standard Adabas source library must be provided to the SYSLIB DD statement in the assembly step:

- Do not concatenate any CICS load libraries in the SYSLIB DD statement when linking the LNCSTUB load module.
- In the SYSLIN data stream after the LNCSTUB object deck, use just the control statement

```
NAME LNCSTUB(R)
```
- Do not include the CICS stub modules DFHEAI0 & DFHEAI1 with the LNCSTUB load module. As a result, however, the following occurs:

- The linkage editor issues IEW462 or similar messages indicating that DFHEAI1 is an unresolved external reference;
- The LNCSTUB module may be marked NOT EXECUTABLE by the linkage editor;
- A condition code of 8 may be set in the link step.

When the application program is linked with LNCSTUB, all the external references are resolved. Use of the link-edit parameters LET and NCAL are recommended so the missing CICS stub pieces result in a condition code of '04' from the link-edit of LNCSTUB.

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 module has either an entry name of LNCSTUB or the alias entry name as coded in the SETC statement to set the value of &STBNAME. Either value may 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/VS. 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.

This section describes each of these IVPs:

- Install and Execute the Assembler IVP: ALCSIVP
- Install and Execute the COBOL IVP: COBSIVP

Note:

The two installation verification programs ALCSIVP and COBSIVP only use fields AA and AE from the Software AG-provided demonstration EMPLOYEES file. For more information about the Software AG-provided demonstration files, read *Load the Demonstration Files* in the z/OS installation instructions.

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 install and execute the Assembler IVP, ALCSIVP:

1. Modify the source member ALCSIVP in ACIvrs.SRCE:

- Edit the file number field DBFNR to be sure it matches the value needed to access the EMPLOYEES file on the Software AG-provided demonstration database you intend to use. For more information about the Software AG-provided demonstration files, read *Load the Demonstration Files* in the z/OS installation instructions.

The ALCSIVP program will take the database-id from the LOGID keyword specified in the ADAGSET macro.

- Check the fields FBUFF, SBUFF and VBUFF for values consistent with your EMPLOYEES file's FDT and data content.
- Check the name used in the EXEC CICS LINK statement to be sure it matches the name of your Adabas CICS command-level link component program. The field LNCNAME is now used and it derives its value from the ENTPT keyword of the ADAGSET macro.

The entry-point alias of the LNCSTUB module can be tested in ALCSIVP by changing the SETC statement for the field &STUBNM to match the entry-point name coded in the LNCSTUB source module using its SETC fieldname &STBNAME.

Note:

The ALCSIVP program will use the value of the ADAGSET keyword PARMTYP to determine whether to use the CICS TWA or CICS COMMAREA to pass data between itself and the Adabas CICS link routine during the first part of its processing when it uses the CICS LINK command to invoke the Adabas CICS link routine. If PARMTYP=TWA is coded in the ADAGSET macro used when ALCSIVP is assembled the CICS TWA is used, otherwise the CICS COMMAREA is used on the EXEC CICS LINK commands.

2. Modify the sample job stream, JCLALCI in ADAvrs.JOBS:

- Member JCLALCI is used to preprocess, assemble, and link the installation verification program ALCSIVP. Place the load module in your CICS DFHRPL library concatenation..
- To modify this JCL to meet your site requirements, change the JOB card in the member and the symbolic values as indicated in the table used in step 1 (see *Step 1, Modify Member JCLLNCS*).

The JCLALCI member uses one additional symbolic parameter: &CICSLIB. This is the name of your CICS RPL library.

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

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

```

DEFINE PROGRAM(ALCSIVP) GROUP(ADABAS)
DESCRIPTION(ADABAS s ASSEMBLER IVP FOR HIGH-PERFORMANCE STUB)
LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(USER) EXECUTIONSET(FULLAPI)

DEFINE TRANSACTION(STB1) GROUP(ADABAS)
DESCRIPTION(TRANSACTION TO EXECUTE THE ASSEMBLER IVP FOR HIGH-PERFORMANCE STUB)
PROGRAM(ALCSIVP) TWASIZE(32) PROFILE(DFHICST) STATUS(ENABLED)
TASKDATALOC(ANY) TASKDATAKEY(USER) STORAGECLEAR(NO)
RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
PRIORITY(1) TRANCLASS(DFHTCLOO) DTIMOUT(NO) INDOUBT(BACKOUT)
RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
RESSEC(NO) CMDSEC(NO)

```

5. 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 install and execute the COBOL IVP, COBSIVP:

1. Modify the source member, COBSIVP in ACI*vrs*.SRCE:
 - Edit the fields WORK-DBID and WORK-FNR to place the desired database ID and file number in the VALUE clauses to access the EMPLOYEES file on the Software AG-provided demonstration database you intend to use. For more information about the Software AG-provided demonstration files, read *Load the Demonstration Files* in the z/OS installation instructions.
 - Ensure that the value in the field LINK-NAME matches the name used in your Adabas CICS command-level link component program.
 - Ensure that the values (literals in the PROCEDURE DIVISION) in the following fields are consistent with the requirements of the EMPLOYEES file FDT and data content you are using:

```

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

```

2. Modify the sample job stream, JCLCOBI in ADA*vrs*.JOBS:
 - Member JCLCOBI is used to preprocess, compile, and link the COBSIVP installation verification program. To modify the JCLCOBI example to meet site requirements, change the JOB card in the member and provide values for the symbolic procedure variables as described in the following table:

Value	Description
&ADALIB	Adabas load library used to provide the ADASTWA load module for the linkage editor.
&MEM	Member name to be processed; in this case, COBSIVP.
&CICSLIB	CICS RPL library where the COBSIVP load module is placed for execution under CICS.
&COBLIB	COBOL compiler STEPLIB.
&INDEX	High-level qualifier for the CICS macro library used in the SYSLIB DD statement for the compiler.
&INDEX2	High-level qualifier for the CICS load library to use for the translator STEPLIB DD statement, and for the SYSLIB in the link step.
&LINKLIB	COBOL LINKLIB.
&STBSRCE	Source library containing the distributed Adabas CICS high-performance stub LNCSTUB.
&STUBLIB	A load library to contain the LNCSTUB load module. This library should be available to your application programs when they are linked.
&SYSMSG	Output class for translator messages.
&SYSOUT	Output class for SYSOUT and SYSPRINT messages.
&WORK	DASD device type to use for temporary and utility data sets.

3. Preprocess, compile, and link COBSIVP:

- Use the modified JCLCOBI 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.

Use the modified JCLCOBI job to preprocess, compile, and link the COBSIVP program. COBSIVP now uses the CICS COMMAREA to pass data to the Adabas CICS link routine, so it is not necessary to link the ADASTWA program with COBSIVP for Version 8.

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.

- Link the COBSIVP program with the LNCSTUB load module and make the LNCSTUB load module available to the linkage editor to be included with the COBSIVP load module.

Note:

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

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

```

DEFINE PROGRAM(COBSIVP) GROUP(ADABAS)
DESCRIPTION(ADABAS s COBOL IVP FOR HIGH-PERFORMANCE STUB)
LANGUAGE(COBOL) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
EXECKEY(USER) EXECUTIONSET(FULLAPI)

DEFINE TRANSACTION(STB2) GROUP(ADABAS)
DESCRIPTION(TRANSACTION TO EXECUTE THE COBOL IVP FOR HIGH-PERFORMANCE STUB)
PROGRAM(COBSIVP) TWASIZE(32) PROFILE(DFHCICST) STATUS(ENABLED)
TASKDATALOC(ANY) TASKDATAKEY(USER) 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)

```

5. 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, COBOL/LE PL/I, and C.

- Use of the CICS Transaction Work Area

A transaction that uses the Adabas DCI or the Adabas CICS command-level link component may 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).

- Use of the CICS COMMAREA

With the Adabas Version 8 CICS link routines and the Adabas 8 LNCSTUB module, use of a CICS COMMAREA to pass data on EXEC CICS LINK commands is strongly recommended. The CICS COMMAREA must be at least 32 bytes in length and the first 8 bytes of the COMMAREA must contain the string "ADABAS52" or "ADABAS8X". The string "ADABAS8X" is for applications that exclusively use the new Adabas Version 8 ACBX direct call interface and its parameter list.

- 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 unless the application program will also be marked as reentrant.

- CICS Requests Issued by LNCSTUB

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

```
EXEC CICS ADDRESS EIB
EXEC CICS LINK
```

If the TWA is used to pass data to the Adabas command-level link:

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

- 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 Adabas parameter list needed to execute an Adabas request. (Either a version 7 or version 8 parameter list may be used)

- 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 Adabas parameter list offsets are summarized in the table below (note that an ACB call is used):

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(LNCSTUB)
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 'LNCSTUB' 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 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 IBM 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
```

The Assembler object input is similar to:

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

When examining the cross-reference from the linkage editor, the symbol “entry-name” 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 SYSLIN input is similar to:

```
REPLACE DFHEAI0
INCLUDE SYSLIB(DFHECI)
```

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 “entry-name” 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 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 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.

Modifying Source Member Defaults (ADAGSET Macro)



Warning:

In Adabas 8, the ADAGSET macro found in the Adabas 8 ACIvrn.SRCE library, should only be used for generating default values for the Adabas 8 CICS high-performance stub routine.

To facilitate the assembly of the Adabas CICS high-performance stub routine, Software AG recommends that you program the ADAGSET macro with site-specific default values and put it in a source library that is available in the SYSLIB concatenation during assembly.

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

- AVB: Adabas VSAM Bridge Support
- ENABNM: Entry Point Name for Program to Enable Adabas TRUE
- ENTPT: Name of the Adabas CICS Command-Level Link Routine
- LOGID: Default Logical Database ID
- LRINFO: Length of Adabas Review Data Area
- LUINFO: Length of User Data passed to Adabas LNKUEXIT1 and LNKUEXIT2
- LUSAVE: Size of User Save Area for Adabas LNKUEXIT1 and LNKUEXIT2
- LXITAA: Length of Work Area provided to LNKUEXIT2
- LXITBA: Length of Work Area for LNKUEXIT1
- MRO: Multiple Region Option
- NETOPT: Method Used to Create User ID
- NTGPID: Natural Group ID
- NUBS: Number of User Blocks Created By CICS Link Routine
- PARMTYP: Area for Adabas Parameter List
- PURGE: Purge Transaction
- RMI: Resource Manager Interface
- SAF: Adabas SAF Security
- SAP: SAP Application Support
- SVCNO: Adabas SVC number

- TRUE: Adabas Task-Related User Exit
- TRUENM: Name of Adabas Task-Related User Exit
- UBLOC: User Block Pool Allocation
- XWAIT: XWAIT Setting for CICS

AVB: Adabas VSAM Bridge Support

Parameter	Description	Syntax
AVB	<p>Indicates whether or not Software AG's Adabas Bridge for VSAM is to be supported by this command-level link routine.</p> <ul style="list-style-type: none"> ● AVB=YES: Adabas VSAM Bridge is to be supported. ● AVB=NO: Adabas VSAM Bridge is not to be supported. 	<pre>AVB={_NO_ YES }</pre>

ENABNM: Entry Point Name for Program to Enable Adabas TRUE

Parameter	Description	Syntax
ENABNM	<p>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. The default value is ADAENAB.</p> <p>This parameter is ignored if TRUE=NO is specified.</p>	<pre>ENABNM={'ADAENAB' 'name' }</pre>

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

Parameter	Description	Syntax
ENTPT	<p>The name given to the Adabas CICS high-performance stub link routine. This name is used in EXEC CICS LINK commands to invoke Adabas services from CICS application programs.</p> <p>See also notes 1 and 2 in the installation procedure.</p>	<pre>ENTPT={'ADACICS' 'name' }</pre>

LOGID: Default Logical Database ID

Parameter	Description	Syntax
LOGID	<p>The value of the default logical database ID. Valid ID numbers are 1-65535.</p>	<pre>LOGID= nnn</pre>

LRINFO: Length of Adabas Review Data Area

Parameter	Description	Syntax
LRINFO	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.	LRINFO={ 0 256 }

LUINFO: Length of User Data passed to Adabas LNKUEXIT1 and LNKUEXIT2

Parameter	Description	Syntax
LUINFO	Length of the user data to be passed from the CICS link routine to Adabas LNKUEXIT1 and LNKUEXIT2. If LUINFO is not specified, the default is zero (no user save area is passed).	LUINFO={ 0 length }

LUSAVE: Size of User Save Area for Adabas LNKUEXIT1 and LNKUEXIT2

Parameter	Description	Syntax
LUSAVE	Size of the user save area to be used by Adabas user exits LNKUEXIT1 and LNKUEXIT2. 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).	LUSAVE={ 0 size }

LXITAA: Length of Work Area provided to LNKUEXIT2

Parameter	Description	Syntax
LXITAA	Length of the work area provided to the LNKUEXIT2 user exit program. Values from 0 (the default) to 32767 may be specified. 0 indicates that no LNKUEXIT2 program is linked with the Adabas command-level link routine and no data is passed to LNKUEXIT2. Note: This parameter is not yet fully implemented. It is provided for future use by the CICS user exit A program linked with LNKOLM.	LXITAA={ 0 nn }

LXITBA: Length of Work Area for LNKUEXIT1

Parameter	Description	Syntax
LXITBA	<p>Length of the work area provided to the LNKUEXIT1 user exit program.</p> <p>Values from 0 (the default) to 32767 may be specified. 0 indicates that no LNKUEXIT1 program is linked with the Adabas command-level link routine and no data is passed to LNKUEXIT1.</p> <p>Note: This parameter is not yet fully implemented. It is provided for future use by the CICS user exit A program linked with LNKOLM.</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> LXITBA={ 0 nn } </div>

MRO: Multiple Region Option

Parameter	Description	Syntax
MRO	<p>The MRO parameter is used to indicate whether or not the CICS multiple region option is to be used.</p> <p>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.</p> <p>If MRO=YES, NETOPT must be set to NETOPT=NO (the default) to prevent non-unique LU names from multiple application regions.</p> <p>If NETOPT=YES and MRO=YES are specified, an assembler MNOTE and a return code of 16 are produced from the assembly step.</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> MRO={ NO YES } </div>

NETOPT: Method Used to Create User ID

Parameter	Description	Syntax
NETOPT	<p>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.</p> <p>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.</p>	<pre>NETOPT={ NO YES }</pre>

NTGPID: Natural Group ID

Parameter	Description	Syntax
NTGPID	<p>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.</p> <p>There is no default value. If no value is specified, the Adabas internal user ID is built in the conventional manner.</p> <p>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.</p> <p>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.</p>	<pre>NTGPID=4-byte-value</pre>

NUBS: Number of User Blocks Created By CICS Link Routine

Parameter	Description	Syntax
NUBS	<p>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.</p> <p>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.</p>	<pre>NUBS={ <u>50</u> <i>blocks</i> }</pre>

PARMTYP: Area for Adabas Parameter List

Parameter	Description	Syntax
PARMTYP	<p>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). When PARMTYP=COM, the Adabas parameters are supplied in the CICS COMMAREA provided by the calling program with the EXEC CICS LINK command. The COMMAREA list for an ACB call 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.</p> <p>PARMTYP=ALL or PARMTYP=COM must be used if the TRUE=YES option is specified.</p>	<pre>PARMTYP={ <u>ALL</u> COM TWA }</pre>

PURGE: Purge Transaction

Parameter	Description	Syntax
PURGE	<p>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.</p> <p>If PURGE=NO (the default) is specified, the NONPURGEABLE option is generated.</p>	<pre>PURGE={ <u>NO</u> YES }</pre>

RMI: Resource Manager Interface

Parameter	Description	Syntax
RMI	<p>The RMI parameter is used to indicate whether or not the CICS Resource Manager Interface is to be used.</p> <p>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).</p> <p>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.</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> RMI={ NO YES } </div>

SAF: Adabas SAF Security

Parameter	Description	Syntax
SAF	<p>Indicates whether or not the Adabas SAF Security (ADASAF) is to be used. If you are using ADASAF, you must set SAF=YES.</p> <ul style="list-style-type: none"> ● YES: Adabas SAF Security is to be used. ● NO: Adabas SAF Security is not to be used. <p>ADASAF requires the Adabas task-related user exit (TRUE) when running under CICS/TS 1.1 or above. When SAF=YES and TRUE=YES, the task-related user exit passes the user's external security ID (sign-on) to Adabas.</p> <p>If TRUE=YES is not specified in this case, the ADAGSET macro terminates the LNKOLSC, LNKTRUE, or LNKENAB assembly process with an MNOTE and a return code of 16.</p> <p>TRUE=YES is not required when running ADASAF under CICS/ESA 3.3 or below. The combination SAF=YES and TRUE=NO is valid in such cases.</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> SAF={ NO YES } </div>

SAP: SAP Application Support

Parameter	Description	Syntax
SAP	<p>The SAP parameter is used to indicate whether or not Adabas support for the SAP application system is required.</p> <p>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.</p> <p>For more information, refer to the supplementary information provided to customers using the SAP application system.</p>	<pre>SAP={ NO YES }</pre>

SVCNO: Adabas SVC number

Parameter	Description	Syntax
SVCNO	The SVCNO parameter is used to specify the value of the Adabas SVC number.	<pre>SVCNO={ 0 nnn }</pre>

TRUE: Adabas Task-Related User Exit

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

TRUENM: Name of Adabas Task-Related User Exit

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

UBPLOC: User Block Pool Allocation

Parameter	Description	Syntax
UBPLOC	<p>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.</p> <p>The ECB used by the EXEC CICS WAIT WAITCICS or the EXEC CICS WAIT EXTERNAL is included in the UB pool.</p> <p>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.</p> <p>Refer to the IBM manual <i>CICS Application Programming Reference</i> for more information.</p>	<pre>UBPLOC= { ABOVE BELOW }</pre>

XWAIT: XWAIT Setting for CICS

Parameter	Description	Syntax
XWAIT	<p>The XWAIT parameter is used to specify whether a standard EXEC CICS WAITCICS (XWAIT=NO) or a WAIT EVENTS EXTERNAL (XWAIT=YES) will be generated into the command-level link component by the assembler process in the LNKOLSC module. XWAIT=YES is the default.</p> <p>The CICS WAIT EVENTS EXTERNAL (XWAIT=YES) is the recommended interface for CICS/TS 1.1 and above.</p> <p>The CICS WAITCICS statement (XWAIT=NO) is provided for use with CICS/MVS 2.1.2 and for CICS/VSE 2.1 through 2.3. It may also be used for CICS/TS 1.1 and above, but may result in poor CICS transaction performance or unpredictable transaction results in busy CICS environments.</p> <p>Note: If XWAIT=NO is specified for use under CICS/ESA 3.3, IBM APAR PN39579 must be applied to the CICS/ESA 3.3 system. For CICS/TS 1.1 and above, this APAR is not required.</p>	<pre>XWAIT={ NO YES }</pre>

Notes:

1. The default for the XWAIT parameter is XWAIT=YES to conform with IBM usage.
2. If XWAIT=NO is specified, the LNKOLSC module issues an EXEC CICS WAITCICS command instead of the EXEC CICS WAIT EVENT command. This conforms with recommended IBM usage of the WAIT and ECB lists in a high-transaction volume CICS system with CICS/TS Version 1.1 and above.

- 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 Posting Mechanisms

CICS WAITCICS (XWAIT=NO) can support a soft post of the specified ECB. This has the disadvantage of becoming a low priority dispatchable unit of work in a CICS/TS environment, since the hand-postable work is not processed by CICS on every work cycle.

EXEC CICS WAIT EXTERNAL (XWAIT=YES), on the other hand, allows CICS to make use of its special post exit code, and will always be checked and processed (if posted) on every CICS work cycle.

For more details on the differences between the various CICS WAIT commands and their relationship to hard and soft posting mechanisms, consult the IBM *CICS/TS Application Programming Reference* and the texts accompanying IBM APAR PN39579 or Item RTA000043874 on the IBM InfoLink service.

XWAIT and the Adabas SVC / Router

The Adabas SVC is fully compatible with the XWAIT=YES setting. The SVC performs the necessary hard post for Adabas callers under CICS/TS using the Adabas 6 command-level link routine. The same SVC performs a soft post for batch callers where the hard post is not required.

Installing Adabas with Com-plete under Adabas 8

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.

Software AG's TP monitor, Com-plete requires an Adabas link routine if it is to communicate with Adabas databases, use Software AG's Entire Net-Work product, or use products like Entire System Server running under Com-plete. At this time, Com-plete does not support a mixed Adabas 7 and Adabas 8 link routine environment; thus Com-plete must be run with either an Adabas 7 link routine or an Adabas 8 link routine.

The Adabas Version 8 link routine is delivered in member ADALCO of the Adabas 8 z/OS load library. This member must be linked with a link globals module you prepare and with any link routine exits you require to create the final ADALCO load module that is loaded by Com-plete when Com-plete is initialized. The final ADALCO load module and any exits linked with it must be reentrant.

The following table lists the modules supplied in your Adabas installation to support the installation of Adabas with Com-plete under Adabas 8.

Module	Description
ADALCO8	Base module
ADALCO	Executable default module

To prepare the Adabas 8 link routine:

- Copy sample member LCOGBL provided in the Adabas 8 ADA vrs .SRCE library to any appropriate user source library where it can be modified (where vrs is the number of the latest Adabas *version* delivered on the tape). LCOGBL is a module containing LGBLSET parameters that are used to

create default settings for command-level link components. A complete description of LGBLSET parameters can be found in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*.

2. Modify the LCOGBL member in the user source library.

At a minimum supply values for the following LGBLSET parameters in LCOGBL:

Parameter	Specify...
LOGID	The default database or target ID. This should be a numeric value between "1" and "65535". The default value is "1". Note: Specifying default values for LOGID and SVCNO under Com-plete is for documentation purposes only. The ADASVC Com-plete runtime control statements will provide the supported database ID/Adabas SVC combinations to be used in running Com-plete. For more information, read the Com-plete documentation.
SVCNO	The default Adabas SVC number. For z/OS, this number should be between "200" and "255". Note: Specifying default values for LOGID and SVCNO under Com-plete is for documentation purposes only. The ADASVC Com-plete runtime control statements will provide the supported database ID/Adabas SVC combinations to be used in running Com-plete. For more information, read the Com-plete documentation.
OPSYS	The three-character abbreviation for the operating system under which Com-plete executes. Valid values include "ZOS" and "VSE". Note: The OPSYS parameter must be set to ZOS.
TPMON	COM. This keyword specifies the three-character TP monitor abbreviation. For Com-plete, this value should be "COM".
RENT	YES. This keyword indicates whether or not the module is serially reentrant. For Com-plete, this value should be "YES".
GEN	CSECT. This keyword indicates whether a CSECT or DSECT is generated. CSECT must be specified so an object module is generated that can be linked as the link routine globals load module.
UES	Whether Adabas Universal Encoding Support (UES) should be enabled. The default is YES. For more information, read <i>Enabling Universal Encoding Support (UES) for Your Adabas Nucleus</i> .

Parameter	Specify...
exit parameters	Whether any other exits are to be active, and in the case of user exits you provide, specify the user exit module names. Specify this information in other parameters of LGBLCOM, as described in <i>Modifying Source Member Defaults (LGBLSET Macro) in Version 8</i> .

3. Modify and run sample job ASMGBLE as described at the top of the job. ASMGBLE can be found in the Adabas 8 ADAvrs.JOBS library. When fully modified, the SET statement in the job should reference the LCOGBL member you prepared in the previous step and the NAME link edit control statement should reference the name specified by the GBLNAME parameter in the LCOGBL member.

Once modified, submit the ASMGBLE job to assemble and link-edit the link globals module.

A new link globals module (with the name specified by the GBLNAME parameter in LCOGBL) will be generated in the user load library identified in the ASMGBLE job.

4. Copy sample job LNKLCO8 to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the ADALCO8 base module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from this job to an appropriate user load library. LNKLCO8 can be found in the Adabas 8 ADAvrs.JOBS library.

The module resulting from this job is called ADALCO.

5. Place the ADALCO module in a load library available in the job step that will start Complete.

The Adabas 8 link routine is prepared.

General Considerations for Installing Adabas with Batch/TSO

When installing Adabas 8 on TSO systems, Adabas-TSO communication is provided by the batch link routines ADALNK8 (non-reentrant) and ADALNKR8 (reentrant).

In this version of Adabas, the ADALNK routines are UES-enabled as distributed. See the section *Enabling Universal Encoding Support (UES) for Your Adabas Nucleus* for more information.

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 all be available from the defined load libraries for that specific TSO user at execution time:

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

This section covers the following topics:

- Non-reentrant ADALNK Batch Routine Operation
- ADALNKR: Reentrant Batch Link Routine

Non-reentrant ADALNK Batch Routine Operation

The ADALNK module in the Adabas 8 load library operates in a Adabas 7-compatible manner when the following conditions are met:

- The calling application must be linked with ADAUSER. If the calling application is not linked with ADAUSER, the ADALNK will not work.
- The ADARUN module from the most recent Adabas 8 load library must be used.
- The database ID and Adabas SVC number must be provided as input through DD statements. Otherwise, the values in the link globals table will override these values.

If all three of these conditions are met, the default database ID and Adabas SVC number will be overridden by the values provided in the DD statement input and passed to the link routine by ADARUN.

Operating in this fashion requires the fewest changes on the part of your data base administrator (DBA) and application programmer. This is also the recommended mode of operation when executing Adabas utilities.

ADALNKR: Reentrant Batch Link Routine

Several Software AG products require the use of a reentrant batch link routine and the ADALNKR load module is provided in the Adabas load library to support them. The Adabas 8 ADALNKR source module is not provided.

You can change default values for these reentrant batch link routines. For more information, read one of the following sections:

- *Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules*
- *Zapping the Default Values for the Adabas 8 ADALNK or ADALNKR Modules*

Software AG recommends that batch application programs be linked with the ADAUSER module, not ADALNK or ADALNKR. The ADAUSER load module is not reentrant, but the reentrant link routine module may be linked with it as long as the application program conforms to the calling requirements described in *Adabas 8 Batch/TSO Reentrant Link Routine (ADALNKR) Calling Requirements* and the PROG=RENTUSER ADARUN parameter is provided in DDCARD input instead of the keyword parameter PROG=USER.

When using the latest Adabas 8 ADALNKR module to obtain reentrant operation under batch or TSO, you must prepare the ADALNKR module in advance. It must be linked with a customized link globals table that provides defaults for the database ID, Adabas SVC number, and other requirements. Any reentrant exits should also be linked with it as required.

Installing Adabas with Batch/TSO under Adabas 8

When installing Adabas 8 on TSO systems, the standard Adabas 8 batch link routine (ADALNK) provides Adabas/TSO communication (SMA job number I056).

This section covers the following topics:

- Supplied Modules
- Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules
- Zapping the Default Values for the Adabas 8 ADALNK or ADALNKR Modules


Supplied Modules

The following table lists the modules supplied in your Adabas installation to support the installation of Adabas with batch/TSO under Adabas 8.

Module	Description
ADALNK8	Base module
ADALNKR8	Base reentrant module
ADALNK	Executable default module
ADALNKR	Executable default reentrant module

Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules

You can change default values for various link routine parameters used by the Adabas 8 ADALNK and ADALNKR modules.

 **To change default values, complete the following steps:**

1. Copy the sample member LNKGBLS (for non-reentrant links) or LNKRGBL (for reentrant links) members provided in the Adabas 8 ADA*vrs* (where *vrs* is the number of the latest Adabas *version* delivered on the tape).SRCE library to any appropriate user source library where they can be modified. These modules contain LGBLSET parameters that are used to create default settings for link components. A complete description of LGBLSET parameters can be found in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*.
2. Modify the LNKGBLS or LNKRGBL member in the user source library. Provide values for the LOGID, SVC, and other keywords to suit your installation requirements.

Note:

The OPSYS parameter must be set to ZOS.

3. Modify and run sample job ASMGBLS as described at the top of the job. ASMGBLS can be found in the Adabas 8 ADA*vrs*.JOBS library. When fully modified, the SET statement in the job should reference the LNKGBLS or LNKRGBL member you prepared in the previous step and the NAME link edit control statement should reference the name specified by the GBLNAME parameter in the LNKGBLS or LNKRGBL member.

Once modified, submit the ASMGCLS job to assemble and link-edit the link globals module.

A new link globals module (with the name specified by the GBLNAME parameter in the LNKGBLS or LNKRGBL member) will be generated in the user load library identified in the ASMGCLS job.

4. Copy sample job LNKLNK8 or LNKLNKR8 (reentrant) to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the appropriate ADALNK8 or ADALNKR8 (reentrant) base module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from the job to an appropriate user load library. LNKLNK8 and LNKLNKR8 can be found in the Adabas 8 ADA*vrs*.SRCE library.

The module resulting from this job is called ADALNK or ADALNKR (as appropriate).

5. Tailor the ADARUN DDCARD input for the job steps that will use the Adabas 8 batch/TSO link routines. The DDCARD input should include the following updates:
 - Specify the ADARUN PROG=USER parameter for a non-reentrant link routine, or specify ADARUN PROG=RENTUSER to use a reentrant link routine in the job step. For more information about the PROG parameter, read *PROGRAM : Program to Run*.
6. Make sure the appropriate load libraries are made available to the job step. These may be STEPLIB, TASKLIB, JOBLIB, or, for reentrant modules, the LPA or LINKLIB.

Zapping the Default Values for the Adabas 8 ADALNK or ADALNKR Modules

You can change default values for various link routine parameters used by the Adabas 8 ADALNK and ADALNKR modules.

Changes to some default values for the Adabas 8 batch/TSO link routines, ADALNK and ADALNKR, may occur with a zap to either the ADALNK or ADALNKR module. This includes the default values for the database ID and the Adabas SVC number. All other default values should be set using the link globals table, as described in *Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules*. Software AG recommends changing all values in the link globals table and relinking ADALNK or ADALNKR (as appropriate).

Use the following IMASPZAP control statements to change default values in ADALNK or ADALNKR (as appropriate):

```

NAME ADALNK LNKGBLS
VER 0030 0001           Default DBID
REP 0030 ####          Site-specific DBID
VER 0032 0AF9          Default Adabas SVC number
REP 0032 0A##          Site-specific Adabas SVC number
*
NAME ADALNKR LNKRGBL
VER 0030 0001           Default DBID
REP 0030 ####          Site-specific DBID
VER 0032 0AF9          Default Adabas SVC number
REP 0032 0A##          Site-specific Adabas SVC number

```

Establishing Adabas SVC Routing by Adabas Database ID

Your application programs that use Adabas link routines in z/OS and VSE environments can route database calls through specific Adabas SVCs, based on the database ID used in the call. SVC routing is managed through the use of a DBID/SVC routing table you supply. Up to 1000 database IDs may be specified in the table and associated with any number of valid SVC numbers installed in the z/OS or VSE system. The DBID/SVC routing table is created using the MDBSVC macro.

Duplicate database IDs are not allowed in the DBID/SVC routing table as there is no reliable way for the link routine to determine which SVC should be used for a database ID if it is listed more than once. If duplicate database IDs are found while the table is being assembled, they are flagged with an assembler MNOTE and a return code of 16 is returned for the assembly attempt.

Notes:

1. Adabas client-based add-ons, such as Adabas Transaction Manager, are not compatible with this feature since for client-based functionality to work, it must be channeled through only a single router for any given session, not across routers. To avoid problems if the dynamic SVC by DBID routing feature is enabled for these products, error messages are issued, the assembly step of the globals table will receive return code 16, and the globals table load module will not be generated.
2. ADALNK linked with the ADASVCTB should only be used by application programs and should not be made available to the Adabas nucleus or to Entire Net-Work.

Caution:

This feature should be used with caution. Transactional integrity is not guaranteed. If an application makes calls to multiple databases that are routed to more than one Adabas SVC, it becomes possible to issue ET, BT, OP, CL, RC, or other Adabas commands that may affect the transaction on one database, but not on the other databases running on different Adabas SVCs that were accessed previously. It therefore is the responsibility of the application program to ensure that all necessary logic is included to ensure transactional integrity across multiple databases where multiple Adabas SVCs are employed.

This section covers the following topics:

- Installing the Adabas DBID/SVC Routing Feature
- General Operation
- Using the MDBSVC Macro

Installing the Adabas DBID/SVC Routing Feature

The general steps for installing the Adabas DBID/SVC routing feature are:

1. Define the DBID/SVC routing table in a library member using MDBSVC macro statements. For more information about the DBID/SVC routing table and the MDBSVC macro, read *Using the MDBSVC Macro*.
2. Assemble and link-edit the DBID/SVC routing table member to create a load module or PHASE that will be made available to the operating environment where the SVC routing feature will be used.

3. Modify a link globals table for the operating environment, specifying the LGBLSET keywords DYNDBSVC=YES and DBSVCTN=*name*, where *name* is the name of the DBID/SVC routing table load module that should be used by the link routine. Assemble and link-edit the updated link globals table as required for the operating environment. For more information about the link globals table and the LGBLSET macro, read *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*. For information on assembling and link-editing the link globals table once the table is updated, refer to the instructions for each z/OS or VSE TP monitoring environment, provided elsewhere in this section.
4. Make the prepared DBID/SVC routing table available in a load library that is accessible by the application program's job step, so it can be loaded by the link routine when it runs.
5. Except for CICS systems, you will need to relink ADALNK or ADALNKR making sure that the INCLUDE statements for the LNKDSL and DEPRTR (or RTRVSE on VSE) modules are included in the job.

This section covers the following topics:

- Installing DBID/SVC Routing under z/OS Batch, TSO and IMS
- Installing DBID/SVC Routing under CICS

Installing DBID/SVC Routing under z/OS Batch, TSO and IMS

The installation steps for the Adabas SVC routing feature under z/OS batch, TSO, and IMS are the same.

 **To install the Adabas DBID/SVC routing feature under z/OS batch, TSO, or IMS, complete the following steps:**

1. Define or modify the DBID/SVC routing table by coding a series of MDBCSVC macros in a library member. Sample member ADASVCTB is provided in the ADAvrs.SRCE library as a template for preparing this member. For more information about using the MDBSVC macro, read *Using the MDBSVC Macro*.
2. Assemble and link-edit the DBID/SVC routing table member to create the table as a load module that you can make available to the application execution job step. The load module should be linked non-reusable and non-reentrant because the link routine subprogram LNKDSL will need to store the addresses of the Adabas SVC IDT headers in the DBID/SVC module to reduce the operating overhead on multiple commands accessing the same Adabas SVC.
3. Define or modify a link globals table for the execution environment. The following LGBLSET keywords are required to support the Adabas SVC routing feature:

LGBLSET Keyword Setting	Description
DYNDBSVC=YES	This keyword and setting indicate that Adabas SVC routing is active for this job step.
DBSVCTN= <i>name</i>	This keyword specifies the name of the DBID/SVC table for this job step. This name must match the name of the load module created to ensure the proper table is loaded when the link routine runs.

- Assemble and link-edit the updated link globals table, as described for the appropriate TP monitor. For batch/TSO, read *Installing Adabas with Batch/TSO under Adabas 8*; for IMS, read *Installing Adabas with IMS TM under Adabas 8*.
- Relink ADALNK or ADALNKR, making sure that the INCLUDE statements for the LNKDSL and DEPRTR modules are included in the job. Samples of the jobs used to relink ADALNK and ADALNKR are listed in the following table:

Link Routine	Sample Job		
	z/OS batch	TSO	IMS
ADALNK	LNKLNK8	LNKLNK8	---
ADALNKR	LNKLNKR8	LNKLNKR8	---
ADALNI8	---	---	LNKLNI8

Installing DBID/SVC Routing under CICS

 **To install the Adabas DBID/SVC routing feature under CICS, complete the following steps:**

- Define or modify the DBID/SVC routing table by coding a series of MDBCSVC macros in a library member. Sample member ADASVCTB is provided in the ADA*vrs*.SRCE library as a template for preparing this member. For more information about using the MDBSVC macro, read *Using the MDBSVC Macro*.
- Assemble and link-edit the DBID/SVC routing table member to create the table as a load module and place it in a library that will be part of the CICS DFHRPL concatenation. The load module should be linked non-reusable and non-reentrant because the link routine subprogram LNKDSL will need to store the addresses of the Adabas SVC IDT headers in the DBID/SVC module to reduce the operating overhead on multiple commands accessing the same Adabas SVC.
- Define the load module as a program to CICS using RDO, or the DFHCSDUP utility. See member DEFADA8 in the ACI*vrs*.SRCE library for sample DFHCSDUP definition statements. The program attributes should be Reload(No), Resident(Yes), Dataloc(Any), and Exekey(CICS).
- Define or modify a link globals table for the execution environment. The following LGBLSET keywords are required to support the Adabas SVC routing feature:

LGBLSET Keyword Setting	Description
DYNDBSVC=YES	This keyword and setting indicate that Adabas SVC routing is active for this job step.
DBSVCTN= <i>name</i>	This keyword specifies the name of the DBID/SVC table for this job step. This name must match the name of the load module created to ensure the proper table is loaded when the link routine runs.

5. Assemble and link-edit the updated link globals table, as described in *Installing Adabas with CICS under Adabas 8* for z/OS installations.

General Operation

When the Adabas SVC routing feature is installed, as described earlier in this section, it is loaded as described below:

- In batch, TSO, or IMS environments, the DBID/SVC routing table is loaded when the link routine initializes if the LGBLSET DYNDBSVC parameter is set to YES in the link globals table. The address of the routing table is kept in the link routine work area for use by all subsequent calls.
- In CICS environments, the Adabas 8 initialization module ADACIC0, normally run during PLTPI processing, loads and validates the DBID/SVC routing table, if the LGBLSET DYNDBSVC parameter was set to YES in the link globals table for the CICS region. The address of the routing table is kept in the global work area associated with the Adabas 8 task-related user exit (TRUE) module, ADACICT, and is made available on each application call to the TRUE by the Adabas command-level module ADACICS/ADADCI.

When an application call is made, the DBID/SVC routing table is searched by the LNKDSL subroutine which is linked with the appropriate link routine for each operating environment. LNKDSL is called after any LUEXIT1 (link routine user exit 1) is invoked, in case the pre-Adabas call user exit modifies the command's database ID for subsequent processing. The call to LNKDSL is made before any monitoring or Adabas Fastpath exits are called, so the monitoring product, such as Adabas Review, Adabas Fastpath, or Adabas Transaction Manager, will perform their processing based on the appropriate Adabas SVC found in the DBID/SVC routing table.

If the database ID associated with a particular call is not found in the DBID/SVC routing table, the default value for the Adabas SVC as specified by the MDBSVC macro's TYPE=INIT parameter is used. If the SVC located is not an Adabas SVC, or if it is not installed on the z/OS system, an Adabas response code of 213 with subcode 16 or 20 is returned to the application. If the calling database is not active for an SVC number, an Adabas response code of 148 (ADARSP148) is returned to the application.

Duplicate database IDs are not allowed in the DBID/SVC routing table as there is no reliable way for the link routine to determine which SVC should be used for a database ID if it is listed more than once. If duplicate database IDs are found while the table is being assembled, they are flagged with an assembler MNOTE and a return code of 16 is returned for the assembly attempt.

Using the MDBSVC Macro

Use the MDBSVC macro to define various aspects of the Adabas DBID/SVC routing table. Several MDBSVC macros are coded together using TYPE=INIT, TYPE=GEN, and TYPE=FINAL keywords to comprise a source module or member. This source module or member is then assembled and link-edited to build the DBID/SVC routing table load module. Sample member ADASVCTB in ADAvrs.SRCE can be used as a template for creating site-specific versions of the DBID/SVC routing table source module. Here is a sample DBID/SVC routing table source member that uses the CSECT name TESTDBT; when the table is assembled, its load module name will be TESTDBT:

```
TESTDBT CSECT
      MDBSVC TYPE=INIT ,SVC=249,DBID=001
      MDBSVC TYPE=GEN ,SVC=237,DBID=( 2,10,21,33,175,1149) ,           X
              DBID2=(100,101,102,13500)
      MDBSVC TYPE=GEN ,SVC=231,DBID=(226,899)
      MDBSVC TYPE=GEN ,SVC=206,DBID=(15,16,69,99,500,12144)
      MDBSVC TYPE=GEN ,SVC=248,DBID=(14,54,111,177,1213,5775)
      MDBSVC TYPE=GEN ,SVC=249,DBID=(17,19,25,35,42,44,61,76)
      MDBSVC TYPE=FINAL
      END
```

When coding keyword values of MDBSVC macro statements, the assembler rules for continuing lines, identifying lists, and providing keyword values must be followed or assembly errors will result. Keywords and values with lists coded as objects of keywords must be separated by commas. There are no positional parameters used with the MDBSVC macro.

The MDBSVC macro can include the following four types of statements, as described in the following table:

MDBSVC Statement Type	Description	Number Allowed
TYPE=INIT	Only one MDBSVC TYPE=INIT statement can be included in the DBID/SVC routing table source member and it must be the first MDBSVC statement in the member. This statement identifies the beginning of the DBID/SVC routing table. The MDBSVC TYPE=INIT statement may also provide the default database ID and Adabas SVC number used for a call.	1
TYPE=GEN	Any number of MDBSVC TYPE=GEN statements can be included in the DBID/SVC routing table source member. These statements specify the lists of Adabas database IDs associated with specific valid Adabas SVC numbers.	any number, as needed.
TYPE=FINAL	Only one MDBSVC TYPE=FINAL statement can be included in the DBID/SVC routing table source member and it must be the last MDBSVC statement in the member before the assembler END statement. This statement identifies the end of the DBID/SVC routing table.	1
TYPE=DSECT	This statement type is reserved for Software AG internal use only. Do not use this statement type.	0

The MDBSVC TYPE=INIT statement can be preceded by a named CSECT statement and named AMODE and RMODE statements. If the CSECT, AMODE, or RMODE statements are included, the name used in them must agree with the name for the DBID/SVC routing table, as coded in the TABNAME parameter on the MDBSVC TYPE=INIT statement and as specified in the DBSVCTN keyword of the LGBLSET macro used for creating the link globals table.

This section covers the following topics:

- MDBSVC TYPE=INIT Syntax
- MDBSVC TYPE=GEN Syntax
- MDBSVC TYPE=FINAL Syntax
- MDBSVC Parameters

MDBSVC TYPE=INIT Syntax

The syntax for the MDBSVC TYPE=INIT statement is:

```
MDBSVC TYPE=INIT [,SVC=svcno] [,DBID=dbid] [,TABNAME={name|ADBSVCT}] [,OPSYS={ZOS|VSE}]
```

The parameters you can code on the MDBSVC TYPE=INIT statement are described in *MDBSVC Parameters*.

MDBSVC TYPE=GEN Syntax

The syntax for the MDBSVC TYPE=GEN statement is:

```
MDBSVC TYPE=GEN [,SVC=svcno] [,DBID=id[, id]...][,DBID2=id[, id]...]
```

The parameters you can code on the MDBSVC TYPE=GEN statement are described in *MDBSVC Parameters*.

MDBSVC TYPE=FINAL Syntax

The syntax for the MDBSVC TYPE=FINAL statement is:

```
MDBSVC TYPE=FINAL
```

No parameters are valid on the MDBSVC TYPE=FINAL statement.

MDBSVC Parameters

The parameters that can be specified on various MDBSVC statements are as follows:

DBID

The DBID parameter can be coded on both the MDBSVC TYPE=INIT and MDBSVC TYPE=GEN statements.

- When specified on the MDBSVC TYPE=INIT statement, it lists the default database ID associated with the SVC specified in the SVC parameter. In this case, only one database ID can be listed in the DBID parameter on a TYPE=INIT statement.
- When specified on a MDBSVC TYPE=GEN statement, it lists the database IDs associated with the SVC specified in the SVC parameter. If more than one database ID is listed, they should be enclosed in parentheses and separated by commas.

Database IDs listed in the DBID parameter must be numeric and must correspond to the IDs of installed Adabas databases. In z/OS environments, database IDs must range from 1 to 65535. The same database ID cannot be specified on multiple MDBSVC statements; they must be unique across all of the DBID and DBID2 statements in the DBID/SVC routing table. Duplicate values are flagged with an MNOTE, which causes the assembly of the DBID/SVC routing table to stop with return code 16.

The following is an example of some DBID parameters on various MDBSVC statements. Note that two MDBSVC statements list database IDs associated with SVC 237. This allows more database IDs to be coded for the same SVC number. Compare the way this is coded to the way the same example is coded for the DBID2 parameter. Both codings produce the same end result.

```
MDBSVC TYPE=INIT,SVC=249,DBID=1
MDBSVC TYPE=GEN,SVC=237,DBID=(2,4,10,16,21,33)
MDBSVC TYPE=GEN,SVC=237,DBID=(175,1149,1221)
MDBSVC TYPE=GEN,SVC=242,DBID=(3,18)
MDBSVC TYPE=FINAL
END
```

DBID2

The DBID2 parameter can be coded only on MDBSVC TYPE=GEN statements. It lists additional database IDs to be associated with an Adabas SVC specified in the SVC parameter. The DBID2 parameter is optional, but when it is specified, it must follow a DBID parameter.

Database IDs listed in the DBID2 parameter must be numeric and must correspond to the IDs of installed Adabas databases. In z/OS environments, database IDs must range from 1 to 65535. The same database ID cannot be specified on multiple MDBSVC statements; they must be unique across all of the DBID and DBID2 statements in the DBID/SVC routing table. Duplicate values are flagged with an MNOTE, which causes the assembly of the DBID/SVC routing table to stop with return code 16.

The following is an example of some MDBSVC statements that includes a DBID2 parameter. Compare the way this example is coded to the way the same example is coded for the DBID parameter. Both codings produce the same end result.

```
MDBSVC TYPE=INIT,SVC=249,DBID=1
MDBSVC TYPE=GEN,SVC=237,DBID=(2,4,10,16,21,33),
      DBID2=(175,1149,1221)
MDBSVC TYPE=GEN,SVC=242,DBID=(3,18)
MDBSVC TYPE=FINAL
END
```


OPSYS

The OPSYS parameter is an optional parameter that can be coded only on the MDBSVC TYPE=INIT statement. This parameter identifies the operating system where the DBID/SVC routing table is assembled. Valid values for the OPSYS parameter are "ZOS" and "VSE"; the default is "ZOS".

PREFIX

The PREFIX parameter can only be coded only on the MDBSVC TYPE=DSECT statement, which is reserved for internal use by Software AG. Do not use this parameter.

SVC

The SVC parameter can be coded on both the MDBSVC TYPE=INIT and MDBSVC TYPE=GEN statements.

- When specified on the MDBSVC TYPE=INIT statement, it specifies the default Adabas SVC number to be used when the calling application provides a database ID that is not found in the DBID/SVC routing table.
- When specified on a MDBSVC TYPE=GEN statement, it specifies the Adabas SVC number to be associated with the Adabas databases identified by the DBID and DBID2 parameters.

The SVC number listed in the SVC parameter must be numeric and must correspond to the SVC number of an installed Adabas SVC. In z/OS environments, the SVC number must range from 200 to 255. Duplicate SVC values can be coded on multiple MDBSVC statements; this allows you to code long lists of database IDs and associate them with the same Adabas SVC.

In the following example, notice that there are two MDBSVC statements for SVC 249. It is the default SVC for the link routine and is also used for database 1, 3, and 18. There are also two MDBSVC statements for SVC 237; the two statements are used to list nine databases associated with SVC 237 (2, 4, 10, 16, 21, 33, 175, 1149, and 1221).

```
MDBSVC TYPE=INIT,SVC=249,DBID=1
MDBSVC TYPE=GEN,SVC=237,DBID=(2,4,10,16,21,33)
MDBSVC TYPE=GEN,SVC=237,DBID=(175,1149,1221)
MDBSVC TYPE=GEN,SVC=249,DBID=(3,18)
MDBSVC TYPE=FINAL
END
```

TABNAME

The TABNAME parameter is an optional parameter that can be coded only on the MDBSVC TYPE=INIT statement. This parameter specifies the name of the DBID/SVC routing table when the source member does not include a separate (and previously coded) CSECT statement. In this case, the name you specify on the TABNAME parameter is used to generate a named CSECT statement and named AMODE and RMODE directives.

The DBID/SVC routing table name that you specify should be between 1 and 8 alphanumeric characters long. In the following example, a DBID/SVC routing table with the name TESTDBT is coded.

```

MDBSVC TYPE=INIT,SVC=249,DBID=1,TABNAME=TESTDBT
MDBSVC TYPE=GEN,SVC=237,DBID=(2,4,10,16,21,33)
MDBSVC TYPE=GEN,SVC=237,DBID=(175,1149,1221)
MDBSVC TYPE=GEN,SVC=249,DBID=(3,18)
MDBSVC TYPE=FINAL
END

```

Modifying Source Member Defaults (LGBLSET Macro) in Version 8

The Adabas 8 LGBLSET macro is used to set default installation values for the Adabas link routines. It is used to prepare an object module which may either be link-edited with the Adabas 8 link routines or provided to the link routines in the job step where they are run. Your Adabas libraries include sample members provided to support the various teleprocessing (TP) monitors in each environment. Each of these sample members may be copied to an appropriate library and modified to provide the necessary customization required for the link routine that is intended to run in a given environment.

The LGBLSET parameter options with their default values (underlined>) are described in the rest of this section:

- ADL: Adabas Bridge for DL/I Support
- AVB: Adabas Bridge for VSAM Support
- CITSNM: Adabas CICS TS Queue Name
- COR: SYSCOR Exit Support
- DBSVCTN: DBID/SVC Routing Table
- DYNDBSVC: DBID/SVC Routing Table
- ENTPT: Name of the Adabas CICS Command-Level Link Routine
- GBLNAME: Name of Link Globals Module
- GEN: Generate CSECT or DSECT
- IDTNAME: BS2000 IDT Common Memory Name
- IDTUGRP: BS2000 Memory Pool User Bound
- LOGID: Default Logical Database ID
- LUIDX: CICS Link User ID Exit Flag
- LUINFO: Length of User Data Passed to Adabas LUEXIT1 and LUEXIT2
- LUIXNAM: CICS Link User ID Generation Exit Name
- LUSAVE: Size of User Save Area for Adabas LUEXIT1 and LUEXIT2

- LX1NAME: User Exit 1 Module Name
- LX2NAME: User Exit 2 Module Name
- MRO: Multiple Region Option
- NETOPT: Method Used to Create User ID
- NTGPID: Natural Group ID
- NUBS: Number of User Blocks Created By CICS Link Routine
- OPSYS: Operating System
- PARMTYP: Area for Adabas Parameter List
- PRE: DSECT Data Prefix
- PURGE: Purge Transaction
- RENT: Reentrant Module Flag
- RETRYX: Retry Command Exit Flag
- REVIEW: Adabas Review Support
- RMI: Resource Manager Interface
- RTXNAME: Command Retry Exit Name
- SAF: Adabas Security Interface Flag
- SAP: SAP Application Support
- SAPSTR: SAP ID String
- SVCNO: Adabas SVC number
- TPMON: Operating Environment
- TRUENM: CICS TRUE Name
- UBPLC: User Block Pool Allocation
- UBSTIME: User Block Scan Time
- UBTYPE: User Block Type
- UES: Universal Encoding Support
- USERX1: User Exit 1 Flag
- USERX2: User Exit 2 Flag

- XWAIT: XWAIT Setting for CICS

ADL: Adabas Bridge for DL/I Support

Parameter	Description	Syntax
ADL	<p>Indicates whether or not the Consistency Interface of Software AG's Adabas Bridge for DL/I is to be supported by this command-level link routine.</p> <ul style="list-style-type: none"> ● ADL=YES: Adabas Bridge for DL/I Consistency Interface is to be supported. ● ADL=NO: Adabas Bridge for DL/I Consistency Interface is <i>not</i> to be supported. 	ADL= { <u>NO</u> YES }

AVB: Adabas Bridge for VSAM Support

Parameter	Description	Syntax
AVB	<p>Indicates whether or not Software AG's Adabas Bridge for VSAM is to be supported by this command-level link routine.</p> <ul style="list-style-type: none"> ● AVB=YES: Adabas Bridge for VSAM is to be supported. ● AVB=NO: Adabas Bridge for VSAM is <i>not</i> to be supported. 	AVB= { <u>NO</u> YES }

CITSNM: Adabas CICS TS Queue Name

Parameter	Description	Syntax
CITSNM	<p>Specifies the 16-byte string that represents the CICS TS queue name for Adabas. The default is "ADACICS".</p>	CITSNM= { <u>ADACICS</u> <i>qname</i> }

COR: SYSCOR Exit Support

Parameter	Description	Syntax
COR	<p>Indicates whether or not Adabas System Coordinator (SYSCOR), Adabas Transaction Manager, and Adabas Fastpath exits are installed and active.</p> <ul style="list-style-type: none"> ● COR=YES: The exits are installed and active. ● COR=NO: The exits are <i>not</i> installed and active. 	COR= { <u>NO</u> YES }

DBSVCTN: DBID/SVC Routing Table

Parameter	Description	Syntax
DBSVCTN	<p>Provides the name of the DBID/SVC routing table that should be used by the link routine during its execution, if any.</p> <p>The routing table name must conform to names for z/OS standard load modules. It is used by a z/OS LOAD macro/SVC during batch, TSO, or IMS operation or by an EXEC CICS LOAD PROGRAM command during CICS operation.</p> <p>If the load module listed is not found, or if it is found to contain invalid header information, user abend U657 is issued in batch, TSO, or IMS environments.</p> <p>If the load module is not defined to CICS or not found in the CICS DFHRPL concatenation, the Adabas CICS link routine environment is not initialized.</p> <p>Note: If the DYNDBSVC parameter is set to NO, this parameter setting is ignored.</p> <p>For more information about SVC routing by database ID in z/OS environments, read <i>Establishing Adabas SVC Routing by Adabas Database ID</i>.</p> <p>Note: Adabas client-based add-ons, such as Adabas Transaction Manager, are not compatible with this feature since for client-based functionality to work, it must be channeled through only a single router for any given session, not across routers. To avoid problems if the dynamic SVC by DBID routing feature is enabled for these products, error messages are issued, the assembly step of the globals table will receive return code 16, and the globals table load module will not be generated.</p>	DBSVCTN={ name ADASVCTB }

DYNDBSVC: DBID/SVC Routing Table

Parameter	Description	Syntax
DYNDBSVC	Indicates whether Adabas SVC routing by database ID should be enabled for the link routine. DYNDBSVC=YES enables Adabas SVC routing by database ID; DYNDBSVC disables it. The default is NO. For more information about SVC routing by database ID in z/OS environments, read <i>Establishing Adabas SVC Routing by Adabas Database ID</i> .	DYNDBSVC={YES NO}

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

Parameter	Description	Syntax
ENTPT	The name given to the Adabas CICS command-level link routine. 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.	ENTPT={ADACICS name}

GBLNAME: Name of Link Globals Module

Parameter	Description	Syntax
GBLNAME	The name of the link globals module.	GBLNAME={LNKGBLS name}

GEN: Generate CSECT or DSECT

Parameter	Description	Syntax
GEN	Indicates whether a CSECT or DSECT is generated.	GEN={CSECT DSECT}

IDTNAME: BS2000 IDT Common Memory Name

Parameter	Description	Syntax
IDTNAME	The common memory pool name of the BS2000 IDT.	IDTNAME=name

IDTUGRP: BS2000 Memory Pool User Bound

Parameter	Description	Syntax
IDTUGRP	Indicates whether the common memory pool is user bound (BS2000)	IDTUGRP={NO YES}

LOGID: Default Logical Database ID

Parameter	Description	Syntax
LOGID	The value of the default target database ID. Valid ID numbers are 1-65535. The default is "1".	LOGID={ <i>nnn</i> <u>1</u> }

LUIDX: CICS Link User ID Exit Flag

Parameter	Description	Syntax
LUIDX	<p>Indicates whether the CICS link user ID user exit is active.</p> <ul style="list-style-type: none"> ● LUIDX=YES: The link user ID user exit is active. ● LUIDX=NO: The link user ID user exit is <i>not</i> active. <p>The actual name of the user exit is provided in the LUIXNAM parameter.</p>	LUIDX={ <u>NO</u> YES}

LUINFO: Length of User Data Passed to Adabas LUEXIT1 and LUEXIT2

Parameter	Description	Syntax
LUINFO	<p>The length of the user data to be passed to target user exit 4. Valid values are numbers from zero (0) through 32,767.</p> <p>If LUINFO is not specified, the default is zero (no user data is passed).</p>	LUINFO={ <u>0</u> <i>length</i> }

LUIXNAM: CICS Link User ID Generation Exit Name

Parameter	Description	Syntax
LUIXNAM	<p>The name of the user ID generation user exit that should be used by the CICS link routine.</p> <p>The exit program must be written in IBM's high-level assembler language. It may issue EXEC CICS commands. It must either be coded reentrant or quasi-reentrant, if it obtains its own DFHEISTG area and changes that to the task-related user exits (TRUEs),</p>	LUIXNAM={ <u>LUIDXIT</u> <i>name</i> }

LUSAVE: Size of User Save Area for Adabas LUEXIT1 and LUEXIT2

Parameter	Description	Syntax
LUSAVE	<p>The size of the user save area to be used by Adabas user exits LUEXIT1 and LUEXIT2. Valid values range from zero (0) through 256. The default is "72".</p> <p>If LUSAVE is not specified, the default is zero (no user save area is passed).</p>	LUSAVE={ <u>72</u> <i>size</i> }

LX1NAME: User Exit 1 Module Name

Parameter	Description	Syntax
LX1NAME	The name of the link user exit 1 module	LX1NAME={ <u>LUEXIT1</u> <i>name</i> }

LX2NAME: User Exit 2 Module Name

Parameter	Description	Syntax
LX2NAME	The name of the link user exit 2 module	LX2NAME={ <u>LUEXIT2</u> <i>name</i> }

MRO: Multiple Region Option

Parameter	Description	Syntax
MRO	<p>Indicates whether or not the CICS multiple region option (MRO) support is required.</p> <p>If you run the CICS command-level link with the CICS MRO, set this to MRO=YES; otherwise, use the default value MRO=NO.</p> <p>If MRO=YES, NETOPT must be set to NETOPT=NO (the default) to prevent non-unique LU names from multiple application regions.</p> <p>If NETOPT=YES and MRO=YES are specified, an assembler MNOTE and a return code of 16 are produced from the assembly step.</p>	MRO={ <u>NO</u> YES }

NETOPT: Method Used to Create User ID

Parameter	Description	Syntax
NETOPT	<p>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 CICS plus the CICS task number.</p> <p>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.</p>	NETOPT={NO YES}

NTGPID: Natural Group ID

Parameter	Description	Syntax
NTGPID	<p>Specifies a four-byte Natural group ID as required for unique Adabas user ID generation in the CICS sysplex 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.</p> <p>There is no default value. If no value is specified, the Adabas internal user ID is built in the conventional manner.</p> <p>Any four-byte alphanumeric value may be specified, but it must be unique for each Adabas command-level link routine running in a CICS sysplex, 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.</p> <p>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.</p>	NTGPID=4-byte-value

NUBS: Number of User Blocks Created By CICS Link Routine

Parameter	Description	Syntax
NUBS	<p>The number of user blocks (UBs) to be created in the user block pool by the CICS link routine. The number of blocks must be large enough to handle the maximum possible number of concurrent Adabas requests.</p> <p>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.</p>	NUBS={ <u>100</u> <i>blocks</i> }

OPSYS: Operating System

Parameter	Description	Syntax
OPSYS	The operating system in use.	OPSYS={ <u>ZOS</u> VSE CMS BS2 }

PARMTYP: Area for Adabas Parameter List

Parameter	Description	Syntax
PARMTYP	<p>The CICS 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).</p> <p>When PARMTYP=COM, the Adabas parameters are supplied in the CICS COMMAREA provided by the calling program with the EXEC CICS LINK command. The COMMAREA list for an ACB call must be at least 32 bytes long and begin with the label "ADABAS52". The COMMAREA list for an ACBX call must be at least 24 bytes long and begin with the label "ADABAS8X". In addition, the last ABD in the COMMAREA list for an ACBX call must be indicated by setting the VL-bit -- in other words, the high bit in the address must be on (X'80').</p> <p>PARMTYP=ALL (the default) uses both the COMMAREA and TWA to pass the Adabas parameters; in this case, the COMMAREA is checked first.</p> <p>We do not recommend that you attempt to map the CICS TWA to the Adabas 8 ACBX direct call. This is because the TWA is of finite size per transaction and because the TWA is not available at CICS startup. We therefore recommend that CICS programs using the Adabas 8 CICS link routines use the COMMAREA only for passing data.</p>	PARMTYP={ <u>ALL</u> COM TWA }

PRE: DSECT Data Prefix

Parameter	Description	Syntax
PRE	The two-byte string to be used as the DSECT data prefix. The default is "LG".	PRE={ <u>LG</u> <i>prefix</i> }

PURGE: Purge Transaction

Parameter	Description	Syntax
PURGE	<p>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.</p> <p>If PURGE=NO (the default) is specified, the NONPURGEABLE option is generated.</p>	PURGE={ <u>NO</u> YES}

RENT: Reentrant Module Flag

Parameter	Description	Syntax
RENT	Indicates whether the globals module is reentrant.	RENT={ <u>NO</u> YES}

RETRYX: Retry Command Exit Flag

Parameter	Description	Syntax
RETRYX	Indicates whether the retry command exit is active.	RETRYX={ <u>NO</u> YES}

REVIEW: Adabas Review Support

Parameter	Description	Syntax
REVIEW	Indicates whether or not Software AG's Adabas Review performance monitor is installed and active. When REVIEW=YES is specified, a work area of 512 bytes is set up for use by Adabas Review.	REVIEW={ <u>NO</u> YES}

RMI: Resource Manager Interface

Parameter	Description	Syntax
RMI	<p>The RMI parameter is used to indicate whether or not the CICS Resource Manager Interface is in use.</p> <p>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).</p> <p>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.</p>	RMI={ <u>NO</u> YES}

RTXNAME: Command Retry Exit Name

Parameter	Description	Syntax
RTXNAME	The name of the command retry exit module.	RTXNAME= { <u>LUEXRTR</u> <i>name</i> }

SAF: Adabas Security Interface Flag

Parameter	Description	Syntax
SAF	Indicates whether Software AG's Adabas SAF Security support is required.	SAF= { <u>NO</u> YES }

SAP: SAP Application Support

Parameter	Description	Syntax
SAP	<p>Indicates whether or not SAP user ID generation is supported.</p> <p>If SAP=YES is specified, the 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.</p> <p>For more information, refer to the supplementary information provided to customers using the SAP application system.</p>	SAP= { <u>NO</u> YES }


SAPSTR: SAP ID String

Parameter	Description	Syntax
SAPSTR	The four-byte SAP ID string to use.	SAPSTR= { ' <u>SAP*</u> ' <i>string</i> }

SVCNO: Adabas SVC number

Parameter	Description	Syntax
SVCNO	<p>The value of the Adabas SVC number.</p> <p>On z/OS systems, valid values range from 200-255 and the default is "249".</p> <p>On z/VSE systems, valid values range from 32-128 and the default is "45".</p>	SVCNO= <i>nnn</i>

TPMON: Operating Environment

Parameter	Description	Syntax
TPMON	<p>The TP monitor operating environment. Valid values should be specified as follows:</p> <ul style="list-style-type: none"> ● Specify "BAT" to use batch. ● Specify "CICS" to use CICS. ● Specify "COM" to use Com-plete. ● Specify "IMS" to use IMS. ● Specify "TSO" to use TSO. ● Specify "UTM" to use UTM. <p> Warning: Be sure to specify a TP monitor operating environment that is supported on the operating system you selected in the OPSYS parameter. In addition, if OPSYS=CMS is specified, the TPMON parameter should not be specified.</p>	TPMON={ <u>BAT</u> CICS COM IMS }

TRUENM: CICS TRUE Name

Parameter	Description	Syntax
TRUENM	Specifies the module name of the Adabas CICS task-related user exit (TRUE). The default is ADACICT.	TRUENM={ <u>ADACICT</u> name }

UBPLOC: User Block Pool Allocation

Parameter	Description	Syntax
UBPLOC	<p>Specifies whether the user block (UB) pool is to be obtained above (the default) or below the 16-megabyte line in CICS.</p> <p>The ECB used by the EXEC CICS WAIT WAITCICS or the EXEC CICS WAIT EXTERNAL is included in the UB pool.</p> <p>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.</p> <p>Refer to the IBM manual <i>CICS Application Programming Reference</i> for more information.</p>	UBPLOC={ <u>ABOVE</u> BELOW}

UBSTIME: User Block Scan Time

Parameter	Description	Syntax
UBSTIME	<p>Specifies the user block (UB) scan time in <i>fat seconds</i>. A <i>fat second</i> is the interval required to change bit-31 of the doubleword set by an STCK instruction. The default is 1800 seconds.</p> <p>This parameter sets the minimum interval at which the Adabas task-related user exit (TRUE) will decide that a user block entry in the user block pool is eligible for release, if (for some reason) the user block entry was not released by normal Adabas CICS processing. Thus, UBSTIME=1800 indicates that a locked user block entry will be released by the Adabas TRUE if more than 1800 fat seconds have elapsed since the user block entry was locked for an Adabas call.</p> <p>The value of UBSTIME should be set higher than the Adabas CT (transaction time) ADARUN parameter. An ADAM93 message indicating either a post failure or a missing 16 call is likely to occur around the time the user block entry is released or prior to the user block entry's release if the Adabas CT timeout value has been exceeded.</p> <p>Note: The Adabas TRUE will not release a user block entry even if the UBSTIME has elapsed if the ECB associated with the locked user block has not been posted. This is to prevent accidental posting of the wrong CICS task by the Adabas SVC.</p>	UBSTIME={seconds 1800}

UBTYPE: User Block Type

Parameter	Description	Syntax
UBTYPE	<p>Identifies the kind of user block (UB) storage the Adabas CICS installation program and Adabas task-related user exit (TRUE) should obtain and use.</p> <p>Valid values are TASK and POOL. POOL is the default. UBTYPE=POOL causes the installation program to obtain a pool of user blocks in CICS storage. This is the classic mechanism used by Adabas CICS link routines.</p> <p>UBTYPE=TASK changes the behavior of the Adabas CICS installation program and Adabas TRUE so they obtain a single user block element, including any required extensions for user data and Software AG products, for each CICS task that invokes the Adabas TRUE. The user block is obtained in CICS shared storage in user-key. It is released when the Adabas TRUE is driven by CICS at the end of the CICS task. The advantage of UBTYPE=TASK is that there is no scan time required to locate and lock a given UB pool element on each Adabas call. The disadvantages of using UBTYPE=TASK are that a CICS GETMAIN must be issued for each CICS task the first time the Adabas TRUE is invoked for the task and that a CICS FREEMAIN must be issued to release the user block storage at the end of the CICS task.</p> <p>The decision to use UBTYPE=TASK should be based on whether your answers to the following questions are "Yes":</p> <ol style="list-style-type: none"> 1. Do the majority of CICS tasks that use this CICS execution unit run for long periods, issuing many Adabas calls within each task? 2. Do the CICS tasks often trip CPU limits set by CICS execution monitoring programs such as those from Omegamon? <p>UBTYPE=POOL should be used if there are problems with CICS storage fragmentation or when most of the Adabas CICS transactions issues a relatively small number of Adabas calls per CICS task.</p> <p>Software AG encourages you to experiment with values for UBTYPE because it is not possible to reliably predict the mix of transactions used at each site or how they call Adabas.</p>	UBTYPE={ <u>POOL</u> TASK}

UES: Universal Encoding Support

Parameter	Description	Syntax
UES	Indicates whether or not Universal Encoding Support (UES) is required.	UES={NO YES}

USERX1: User Exit 1 Flag

Parameter	Description	Syntax
USERX1	Indicates whether or not user exit 1 is active.	USERX1={NO YES}

USERX2: User Exit 2 Flag

Parameter	Description	Syntax
USERX2	Indicates whether or not user exit 2 is active.	USERX2={NO YES}

XWAIT: XWAIT Setting for CICS

Parameter	Description	Syntax
XWAIT	<p>Indicates whether a standard EXEC CICS WAITCICS (XWAIT=NO) or a WAIT EVENTS EXTERNAL (XWAIT=YES) will be executed by the Adabas 8 task-related user exit (TRUE). XWAIT=YES is the default.</p> <p>The CICS WAIT EVENTS EXTERNAL (XWAIT=YES) is the recommended interface for CICS/TS 1.1 and above.</p> <p>The CICS WAITCICS statement (XWAIT=NO) is provided for use with CICS/MVS 2.1.2 and for CICS/VSE 2.1 through 2.3. It may also be used for CICS/TS 1.1 and above, but may result in poor CICS transaction performance or unpredictable transaction results in busy CICS environments.</p> <p>Note: If XWAIT=NO is specified for use under CICS/ESA 3.3, IBM APAR PN39579 must be applied to the CICS/ESA 3.3 system. For CICS/TS 1.1 and above, this APAR is not required.</p>	XWAIT={NO YES}

Notes:

1. If XWAIT=NO is specified, the ADACICT (Adabas 8 TRUE) module issues an EXEC CICS WAITCICS command instead of the EXEC CICS WAIT EVENT command. XWAIT=YES conforms with recommended IBM usage of the WAIT and ECB lists in a high-transaction volume CICS system with CICS/TS Version 1.1 and above.

2. All EXEC CICS commands are processed by the CICS preprocessor; the LGBLSET parameters cause the subsequent assembly step to skip some of the statements.

XWAIT Posting Mechanisms

CICS WAITCICS (XWAIT=NO) can support a soft post of the specified ECB. This has the disadvantage of becoming a low priority dispatchable unit of work in a CICS environment, since the hand-postable work is not processed by CICS on every work cycle.

EXEC CICS WAIT EXTERNAL (XWAIT=YES), on the other hand, allows CICS to make use of its special post exit code, and will always be checked and processed (if posted) on every CICS work cycle.

For more details on the differences between the various CICS WAIT commands and their relationship to hard and soft posting mechanisms, consult the IBM *CICS Application Programming Reference* and the texts accompanying IBM APAR PN39579 or “Item RTA000043874” on the IBM InfoLink service.

XWAIT and the Adabas SVC / Router

The Adabas SVC is fully compatible with the XWAIT=YES setting. The SVC performs the necessary hard post for Adabas callers under CICS using the Adabas command-level link routine. The same SVC performs a soft post for batch callers where the hard post is not required.