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 z/VSE
- General Considerations for Installing Adabas with CICS
- Installing Adabas with CICS under Adabas 8
- Installing Adabas with Com-plete under Adabas 8
- Installing Adabas with Batch 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 z/VSE

This section covers the following topics:

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

High-Level Assembler

IBM has dropped support for the old z/VSE assembler and Software AG supports assembling the Adabas 7 link components with the high-level assembler only.

Addressing Mode Assembly Directives

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

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

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

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

Software AG recommends RMODE 24 for the z/VSE non-reentrant batch link routine (ADALNK.PHASE).

Modifying the Assembly Directives

These directives may be changed by modifying the source members before assembling them, or they may be overridden by linkage editor control statements. For example, to link the batch ADALNK module with AMODE31 and an RMODE ANY, the following control statements may be provided as input to the linkage editor:

```
PHASE ADALNK,*
MODE AMODE(31),RMODE(ANY)
```

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

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

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 V8 links run AMODE(31), but the Dataloc RDO parameter governs the AMODE and RMODE of the running CICS transaction.

The batch z/VSE link routine, ADALNK, has been assembled and link-edited AMODE(31,RMODE(24). This provides the most flexible configuration for most z/VSE applications that will invoke it. It may be re-linked AMODE(31),RMODE(ANY), but you must be certain that no AMODE(24) applications will invoke it.

The reentrant batch link routine, ADALNKR, has been assembled AMODE(31),RMODE(24). It may be re-linked AMODE(31),RMODE(ANY) if no AMODE(24) applications will invoke it.

The z/VSE Com-plete link routine, ADALCO, has been assembled and link-edited AMODE(31),RMODE(24), and this is the required configuration for ADALCO under z/VSE Com-plete because ADALCO still uses z/VSE macros and services which require it to reside below the 16-megabyte line.

All of the Adabas 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.

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

For prior versions of Adabas, UES is enabled by default for only the batch and Com-plete link routines. As of Adabas version 8, UES is enabled by default for *all* link routines, including the CICS link routines. It is not necessary to disable UES support. Applications that do not require UES translation continue to work properly even when the UES components are linked with the Adabas link routines. See the section *Enabling Universal Encoding Support (UES) for Your Adabas Nucleus* for more information.

This section covers the following topics:

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 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 load module LNKUESL delivered with earlier levels of Adabas Version 7 is no longer supplied since the link jobs now specify the LNKUES or LNKUES7 module and the translation tables separately.

3. The LNKUES module is functionally reentrant; however, they is not linked that way in the Adabas load library.
4. 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.
5. 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

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 LUEXIT1. In Adabas 8 replies, LNKUES receives control after LUEXIT2.

Adabas 8 Jobs for z/VSE Universal Encoding Support

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

Sample Job	Description
ALNKCIC8.X	Assembles and links the CICS globals table with LNKUES and the default translation tables ASC2EBC and EBC2ASC.
ALNKLCO8.X	Relinks the Com-plete link routine with the LCOGBL link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.
ALNKLNK8.X	Relinks the batch link routine with the LNKGBLS link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.
ALNKLNR8.X	Relinks 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 power statements, provide the proper names for the procedures and libraries referenced, and all necessary extent and volume information. Refer to the comments in the jobs themselves for more information.

Disabling UES Support for Adabas 8 Link Routines

This section describes how to disable UES support in the Adabas 8 Com-plete and batch 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 any other keyword 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 Com-plete under Adabas 8*, and *Installing Adabas with Batch under Adabas 8*.

General Considerations for Installing Adabas with CICS

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

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

- The command-level link from Adabas 8 cannot be used with CICS/VSE 2.3. Instead, you must use the command-level link routine for Adabas Version 7.4.4 or the macro-level link routine provided in source in the 7.4.4 VSE sublibrary with CICS/VSE 2.3 environments.

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

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

This section covers the following topics:

- CICS Release Support
- CICS MRO Environment Requirements
- Sample Resource Definitions
- Requirement for CICS Command Resource Security

CICS Release Support

IBM has not announced a date for end of service for CICS/VSE 2.3. Consequently, Software AG will continue to support the Adabas CICS 7.4.4 link routines, particularly the macro-level ADALNC routine on systems running CICS/VSE 2.3 until IBM drops support for that level of CICS.

For CICS/TS 1.1 and above for VSE, the Adabas 8 CICS link components are supported and the Adabas CICS 7.4 link routines will not be supported when general support for Adabas 7.4.4 is terminated by Software AG.

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 LGBLSET option MRO=YES and use the default value for the LGBLSET NETOPT option.

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

Alternatively, you can create a link user exit 1 (LUEXIT1) for the link routine that

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

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

Sample Resource Definitions

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

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 (ADA*vrn*.LIBR).

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
- 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 Under Adabas 8

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 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.X) 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 (see the MACIOPT macro assembly job, ASMCOPT.X).

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/VSE sublibrary.

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	Specifies the name of the link globals table associated with this Adabas CICS stub. This parameter is required. 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).	There is no default.
TRUENAME	Specifies the name of the Adabas CICS task-related user exit (TRUE) to be invoked by this Adabas CICS stub. This parameter is required. 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	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 ADA*vrs* sublibrary as member MACIOPT.A on z/VSE systems. A sample ACIOPT source member is provided in the ADA*vrs* sublibrary on z/VSE systems.

The syntax of the MACINS macro is shown below:

```
MACIOPT ENTRY = GLOBAL,GEN = { CSECT | DSECT }
                       ,CNAME = { ACIOPT | module-name }
                       ,IMSGDEST = { 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	Identifies the destination type for the installation progress and error messages produced by ADACIC0: console, transient data queue, or both. 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. 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. IMSGDEST=BOTH causes installation progress messages to be written both to the console and to a named CICS transient data queue.	CONSOLE

Parameter	Description	Default
IMQNAME	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. 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. The queue may be defined using the CICS RDO facility (using the CEDA transaction) or using the DFHDCT macro. On z/VSE systems, the transient data queue must be defined using the DFHDCT macro. A sample member, DCTACI.A, is provided in the z/VSE ADA*vrs* sublibrary. For more information, consult the appropriate IBM CICS documentation. 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.	There is no default.
MNTRUE	Specifies a maximum value for the number of Adabas CICS execution units (and thus globals tables) to be installed for this CICS or CICSplex. If this number is exceeded, a warning MNOTE and condition code of 4 is produced by the assembler. 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.	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	Specifies the name of the link globals table to be loaded and used to install an Adabas CICS execution unit. This parameter is required. Only one GTNAME parameter can be specified on each ENTRY=GROUP statement.	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

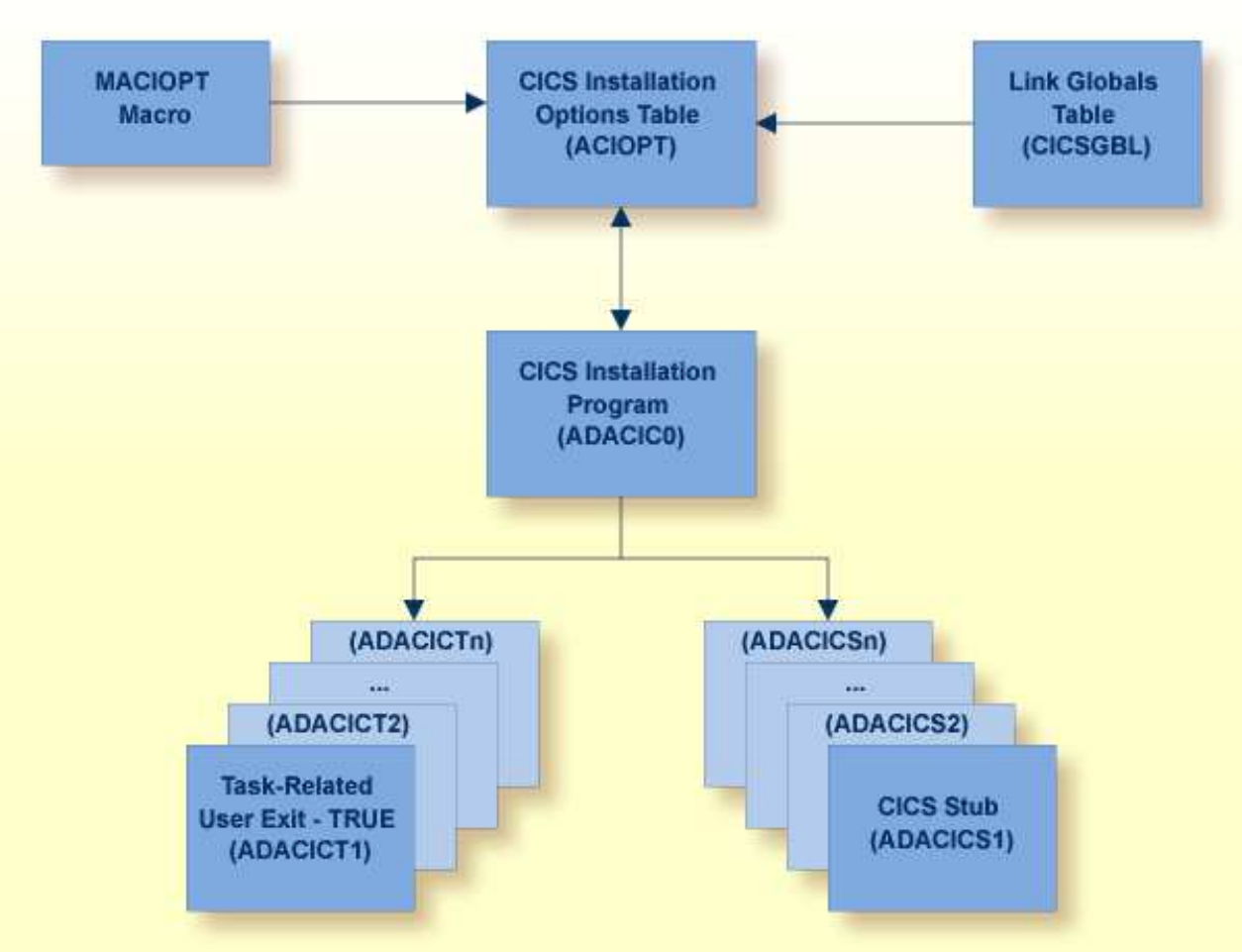
If assembled and link-edited, the following source member will produce the load module ACIOPT and will install two Adabas CICS execution units. One will load a globals table named LNKCI02 and the other will load a globals table named CICSGBL. Installation messages will be written to the CICS transient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

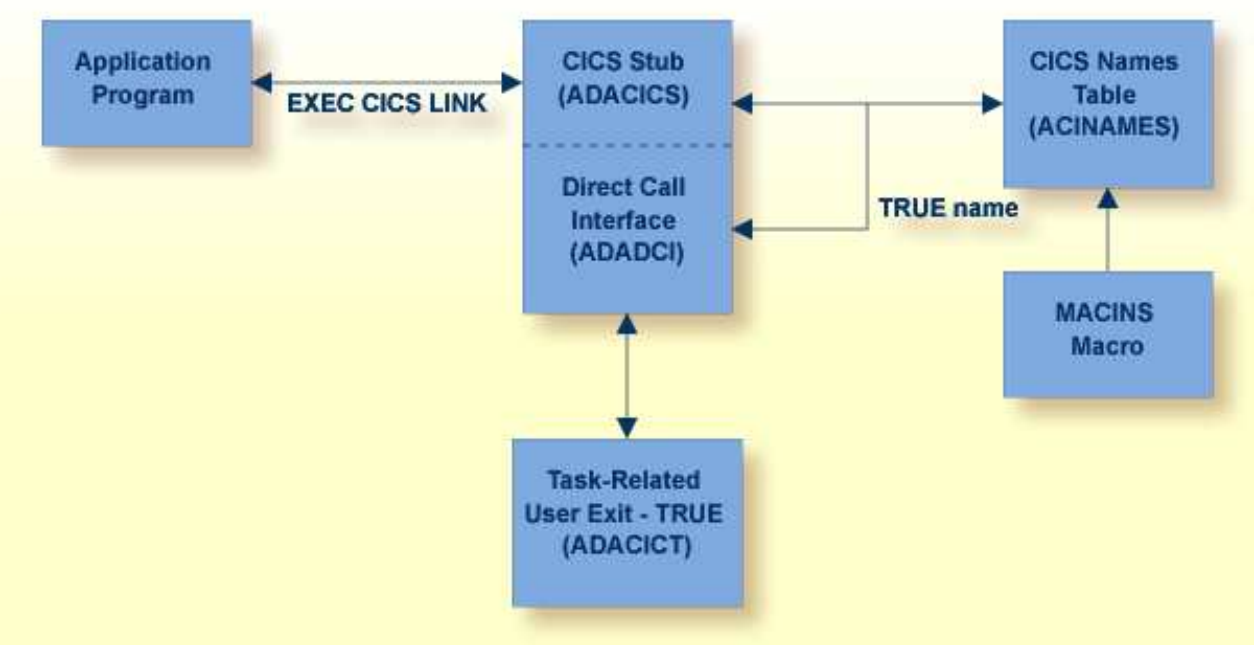
Adabas Task-Related User Exits (TRUEs)

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 8 with CICS TP monitors.

Note:
The Adabas 8 installation supports Adabas 7 direct calls in addition to Adabas 8 calls; however, an Adabas 7 installation does not support Adabas 8 direct calls.

Module	Description
ADACIC0.OBJ	CICS initialization program code object module.
ADACIC0.PHASE	CICS initialization executable module.
ADACICS.OBJ	CICS TP monitor program code object module. This module is linked with ADADCI.OBJ to produce ADACICS.PHASE.
ADACICS.PHASE	CICS TP monitor executable module.
ADACICT.OBJ	CICS task-related user exit (TRUE) program code object module, dependent part. This module is linked with LNKCIM.OBJ to produce ADACICT.PHASE.
ADACICT.PHASE	CICS TRUE executable module.
ADADCI.OBJ	Direct call interface program code object module. This module is linked with ADACICS.OBJ to produce ADACICS.PHASE.
CICSGBL.A	Sample link globals table. This module is modifiable. Once it is modified, you can use the ALNKCIC8.X sample JCS to assemble the CICSGBL.A module, producing the CICSGBL.OBJ object module and then link-editing all relevant CICS program code object modules to create the relevant CICS phases required for Adabas 8 support.
CICSGBL.OBJ	Link globals table object module.
CICSGBL.PHASE	Link globals table executable module.
LNKCIC0.OBJ	CICS link book used when applying maintenance with MSHP to link-edit ADACIC0.OBJ to produce ADACIC0.PHASE.
LNKCICG.OBJ	CICS link book used when applying maintenance with MSHP to link-edit the CICSGBL.OBJ globals table and produce CICSGBL.PHASE.
LNKCICS.OBJ	CICS link book used when applying maintenance with MSHP to link-edit ADADCI.OBJ and ADACICS.OBJ to produce ADACICS.PHASE.
LNKCICT.OBJ	CICS link book used when applying maintenance with MSHP to link-edit ADACICT.OBJ and LNKCIM.OBJ to produce ADACICT.PHASE.
LNKCIM.OBJ	CICS task-related user exit (TRUE) product code object module, independent part. This module is linked with ADACICT.OBJ to produce ADACICT.PHASE.

Installation Procedure Under Adabas 8

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

- Step 1. Modify the CICS Startup JCS
- 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.A)
- Step 7. Assemble and Link-edit the CICS Link Globals Table (ALNKCIC8.X)
- Step 8. Modify CICS Installation Values (DEFADAC.A)
- Step 9. Update the CICS CSD File (DFHCSDUP)
- Step 10. Modify, Assemble and Link the CICS PLTPI Table for ADACIC0
- Step 11. Update, Assemble and Link-edit the Destination Control Table (DCTACI.A)
- Step 12. Start the CICS

Step 1. Modify the CICS Startup JCS

Modify the CICS startup JCS to include the Adabas 8 sublibrary in the LIBDEF chain. This includes the phases ADACIC0, ADACICS, ACACICT and any renamed versions of ADACICS or ADACICT.

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. A sample job, ASMCOPT.X , is provided.

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 ADA*vrs* sublibrary as member MACIOPT.A on z/VSE systems. For complete information on the MACIOPT macro, read *The MACIOPT Macro*, elsewhere in this section.

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" 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 ADA*vrs* sublibrary on z/VSE 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. A sample job, LNKATRU.X , is provided.

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 PHASE statement giving a new name for each Adabas TRUE.

2. Define each named Adabas TRUE as a program to CICS.

Example

For example, the following link-edit control statements would create an Adabas TRUE called "ADATRUE":

```
  PHASE ADATRUE,*
  MODE AMODE(31),RMODE(ANY)
  INCLUDE DFHEAI
  INCLUDE ADACICT
  INCLUDE LNKCIM
  INCLUDE LNKDSL
  INCLUDE RTRVSE
  ENTRY ADACICT
// EXEC LNKEDT ...
```

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/VSE sublibrary.

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,                                    X
               GTNAME=CICSGBL
```

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.

A sample job, ASMCINS.X , is provided.

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).

Example

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

```
  PHASE ADABAS,*
  MODE AMODE(31),RMODE(ANY)
  INCLUDE DFHEAI
  INCLUDE ADACICS
  INCLUDE ADADCI
  INCLUDE ACINAMES
  ENTRY ADACICS
// EXEC LNKEDT ...
```

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.A)

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.A member found in the Adabas 8 ADA*vrs* sublibrary. 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 setting for the OPSYS parameter must be set to "VSE".

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 "VSE".

   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.A member with a unique name in an appropriate user sublibrary.

Step 7. Assemble and Link-edit the CICS Link Globals Table (ALNKCIC8.X)

Using sample job ALNKCIC8.X, assemble and link-edit the member you saved in the previous step into a sublibrary that will be made available to CICS in the LIBDEF concatenation. Note that any user or Software AG link routine exits should be link-edited with this load module. (For information about specific Software AG product exits, read the installation documentation for the product.)

Step 8. Modify CICS Installation Values (DEFADAC.A)

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

Step 9. Update the CICS CSD File (DFHCSDUP)

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

Step 10. 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 ADA*vrn*.LIBR 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.

Assemble and link the modified PLTPI table into a library that will be available to the desired CICS region.

Step 11. Update, Assemble and Link-edit the Destination Control Table (DCTACI.A)

Update a Destination Control Table (DCT) to include the entries found in member DCTACI.A in the Adabas 8 sublibrary. Assemble and link-edit this table with a unique suffix into a sublibrary that will be made available to CICS. Modify the CICS SIT parameters to reference the updated DCT.

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 Adabas with Com-plete under Adabas 8

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

Note:
The Adabas 8 installation supports Adabas 7 direct calls in addition to Adabas 8 calls; however, an Adabas 7 installation does not support Adabas 8 direct calls.

Supplied Module	Description
ADALCO.PHASE	Com-plete TP monitor executable module.
LCOGBL.A	Link globals table source module. This module is modifiable. Once it is modified, you can use sample job ALNKLCO8.X to assemble an LCOGBL.OBJ module . This sample job also link-edits the LCOGBL.OBJ module with LCOVSE8.OBJ to produce ADALCO.PHASE.
LCOGBL.OBJ	Link globals table object module assembled from LCOGBL.A.
LCOVSE8.OBJ	Com-plete TP monitor program code object module.
LNKLCO.OBJ	Com-plete link book containing link-edit control cards used when applying maintenance with MSHP to link-edit LCOGBL.OBJ and LCOVSE8.OBJ to produce ADALCO.PHASE.

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 sublibrary. 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.

To prepare the Adabas 8 link routine:

1. Edit the LCOGBL.A member in the Adabas 8 distribution sublibrary. LCOGBL.A is a module containing LGBLSET parameters that are used to create default settings for Com-plete link components. A complete description of LGBLSET parameters can be found in *Modifying Source Member Defaults (LGBLSET Macro) in Version 8*.

   Note:
   The OPSYS parameter must be set to "VSE".

2. Modify and run the ALNKLCO8.X member to assemble and link-edit the link globals table you updated in the previous step.

   The ALNKLCO8.X member will assemble and catalog the link globals table for Com-plete and link it with the Com-plete link routine, LCOVSE8.OBJ and any required exits. The ALNKLCO8.X member provides link-edit control cards for the inclusion of the Adabas 8 LNKUES module with the ASC2EBC and EBC2ASC translation tables.

3. Place the final phase, ADALCO, in a library that will be part of the Com-plete LIBDEF search chain.

   Note:
   The defaults set in the link globals table for Com-plete are primarily for documentation purposes. The Adabas/Com-plete interface module, TLOPADAB, sets values for Adabas target ID and SVC number on each Adabas call. However, it is necessary to include the link globals table object module and any necessary exits, including user exits when linking the Adabas 8 ADALCO.PHASE. If user exits are to be linked with ADALCO, be certain to code the LGBLSET keywords accordingly.

   The Adabas 8 link routine is prepared.

Installing Adabas with Batch under Adabas 8

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

Batch applications should be linked with the ADAUSER module to provide the greatest degree of application calling isolation when invoking the Adabas batch link routines. The ADAUSER module will provide code to load the appropriate link routine and the supporting ADARUN and ADAIOR modules. ADARUN, in turn, loads other modules. To start a user program linked with ADAUSER, the following

modules must be available in the LIBDEF search chain: ADAIOR, ADAIOS, ADALNK, ADAMLF, ADAOPD, ADAPRF, and ADARUN. In addition, ADAUSER reads DDCARD input from SYSIPT or DISK to allow jobstep setting of the database ID, Adabas SVC number, and other parameters.

For non-reentrant operation, the DDCARD input should provide the keyword PROG=USER. This causes ADARUN to load ADALNK for non-reentrant batch operations.

If you want to use reentrant batch operations, the ADAUSER module can still be linked with the application program, but the PROG=RENTUSER keyword must be coded on the DDCARD input. ADAUSER is, however, non-reentrant. For full reentrant batch applications, it will either need to be loaded (CDLOAD) separately, or the ADALNKR.PHASE must be loaded without using the ADAUSER module. In this case, the default values for DBID, SVC number, length of user information, and which exits are to be used is provided by the linked link globals table, as modified (read *Installing the Reentrant Batch z/VSE Adabas 8 Link Routine*. It is also possible to zap the ADALNKR.PHASE or LNKVSER8.OBJ module with these defaults, but Software AG recommends coding and linking the link globals table instead. Additional information on using a reentrant batch link routine is also provided in *Required Application Reentrancy Properties*.

This section covers the following topics:

- Supplied Modules
- Installing the Batch z/VSE Adabas 8 Link Routine
- Installing the Reentrant Batch z/VSE Adabas 8 Link Routine

Supplied Modules

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

Note:
The Adabas 8 installation supports Adabas 7 direct calls in addition to Adabas 8 calls; however, an Adabas 7 installation does not support Adabas 8 direct calls.

Module	Description
ADALNK.PHASE	Batch executable module.
ADALNKR.PHASE	Batch reentrant executable module.
LNKGBLS.A	Batch link globals table. This module is modifiable. Once it is modified, you can use the ALNKLCO8.X sample JCS to assemble the LNKGBLS.A module, producing the LNKGBLS.OBJ module and then link-editing the LNKGBLS.OBJ module with the LNKVSE8.OBJ module to create the ADALNK.PHASE.
LNKGBLS.OBJ	Batch link globals table object module assembled from LNKGBLS.A.
LNKRGBL.A	Batch reentrant link globals table. This module is modifiable. Once it is modified, you can use the ALNKLNR8.X sample JCS to assemble the LNKRGBL.A module, producing the LNKRGBL.OBJ module and then link-editing the LNKRGBL.OBJ module with the LNKVSER8.OBJ module to create the ADALNKR.PHASE.
LNKRGBL.OBJ	Batch reentrant link globals table object module assembled from LNKRGBL.A.
LNKLNK.OBJ	Batch link book containing link-edit control cards used when applying maintenance with MSHP to link-edit LNKGBLS.OBJ and LNKVSE8.OBJ modules to produce ADALNK.PHASE.
LNKLNKR.OBJ	Batch link book containing link-edit control cards used when applying maintenance with MSHP to link-edit reentrant LNKRGBL.OBJ and LNKVSER8.OBJ modules to produce ADALNKR.PHASE.
LNKVSE8.OBJ	Batch program code object module.
LNKVSER8.OBJ	Batch reentrant program code object module.

Installing the Batch z/VSE Adabas 8 Link Routine

To install the Adabas 8 non-reentrant link routine for z/VSE batch, complete the following steps:

1. Edit member LNKGBLS.A in the Adabas distribution sublibrary. Provide values for the LOGID, SVC, GBLNAME, and other keywords to suit your installation requirements. This module contains LGBLSET parameters 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*.

   Note:
   The OPSYS parameter must be set to "VSE".

2. Edit the ALNKLNK8.X member found in the Adabas 8 sublibrary. This member will assemble and catalog the LNKGBLS.A module and link it and any desired exits with the LNKVSE8.OBJ module to create the ADALNK.PHASE member for Adabas 8. The ALNKLNK8.X member includes sample link-edit control cards to support UES by including the LNKUES.OBJ. module with the ASC2EBC and EBC2ASC translation tables. Modify the link-edit control cards to include any additional

Software AG exit or user exit, as specified in the updated LNKGBLS.A member.

3. Provide the ADALNK.PHASE member in the LIBDEF search chain for the jobstep that will require Adabas database access or Software AG services.

Installing the Reentrant Batch z/VSE Adabas 8 Link Routine

To install the Adabas 8 reentrant link routine for z/VSE batch, complete the following steps:

1. Edit member LNKRGBL.A in the Adabas distribution sublibrary. Provide values for the LOGID, SVC, GBLNAME, and other keywords to suit your installation requirements. This module contains LGBLSET parameters 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*.

   Note:
   The OPSYS parameter must be set to "VSE".

2. Edit the ALNKLNR8.X member found in the Adabas 8 sublibrary. This member will assemble and catalog the LNKRGBL.A module and link it and any desired exits with the LNKVSER8.OBJ module to create the ADALNKR.PHASE member for Adabas 8. The ALNKLNR8.X member includes sample link-edit control cards to support UES by including the LNKUES.OBJ. module with the ASC2EBC and EBC2ASC translation tables. Modify the link-edit control cards to include any additional Software AG exit or user exit, as specified in the updated LNKRGBL.A member.

3. Provide the ADALNKR.PHASE member in the LIBDEF search chain for the jobstep that will require Adabas database access or Software AG services.

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 in 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 z/VSE Batch
- 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 ADA*vrs*.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=*name*	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.

4. 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* .

5. 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 z/VSE Batch

To install the Adabas DBID/SVC routing feature under z/VSE batch, 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.A is provided in the sublibrary SAGLIB.ADA*vrs* 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 PHASE that you can make available to the application execution job step. The PHASE 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=*name*	This keyword specifies the name of the DBID/SVC table for this job step. This name must match the name of the PHASE created to ensure the proper table is loaded when the link routine runs.

4. Assemble and link-edit the updated link globals table, as described for the appropriate TP monitor. For batch/TSO, read *Installing Adabas with Batch under Adabas 8*.

5. Relink ADALNK.PHASE or ADALNKR.PHASE, making sure that the INCLUDE statements for the LNKDSL and RTRVSE object 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
ADALNK.PHASE	ALNKLNK8.X
ADALNKR.PHASE	ALNKLNR8.X

Installing DBID/SVC Routing under CICS

To install the Adabas DBID/SVC routing feature under CICS, 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 ADA*vrs*.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 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.

3. Define the load module as a program to CICS using RDO, or the DFHCSDUP utility. See member DEFADA8 in the ACI*vrs*.SRCE libarary for sample DFHCSDUP definition statements. The program attributes should be Reload(No), Resident(Yes), Dataloc(Any), and Execkey(CICS).

4. 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=*name*	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*Installing Adabas with CICS under Adabas 8* for z/VSE 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 ADA*vrs*.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),                         X
       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
- UBPLOC: 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	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. • ADL=YES: Adabas Bridge for DL/I Consistency Interface is to be supported. • ADL=NO: Adabas Bridge for DL/I Consistency Interface is *not* to be supported.	`ADL={NO|YES}`

AVB: Adabas Bridge for VSAM Support

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

CITSNM: Adabas CICS TS Queue Name

Parameter	Description	Syntax
CITSNM	Specifies the 16-byte string that represents the CICS TS queue name for Adabas. The default is "ADACICS".	`CITSNM={ADACICS|`*`qname`*`}`

COR: SYSCOR Exit Support

Parameter	Description	Syntax
COR	Indicates whether or not Adabas System Coordinator (SYSCOR), Adabas Transaction Manager, and Adabas Fastpath exits are installed and active. ● COR=YES: The exits are installed and active. ● COR=NO: The exits are *not* installed and active.	`COR={NO|YES}`

DBSVCTN: DBID/SVC Routing Table

Parameter	Description	Syntax
DBSVCTN	Provides the name of the DBID/SVC routing table that should be used by the link routine during its execution, if any. 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. 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. 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. **Note:** If the DYNDBSVC parameter is set to NO, this parameter setting is ignored. For more information about SVC routing by database ID in z/OS environments, read *Establishing Adabas SVC Routing by Adabas Database ID*. **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 in 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.	`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 *Establishing Adabas SVC Routing by Adabas Database ID*.	`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={nnn | 1}`

LUIDX: CICS Link User ID Exit Flag

Parameter	Description	Syntax
LUIDX	Indicates whether the CICS link user ID user exit is active. • LUIDX=YES: The link user ID user exit is active. • LUIDX=NO: The link user ID user exit is *not* active. The actual name of the user exit is provided in the LUIXNAM parameter.	`LUIDX={NO|YES}`

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

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

LUIXNAM: CICS Link User ID Generation Exit Name

Parameter	Description	Syntax
LUIXNAM	The name of the user ID generation user exit that should be used by the CICS link routine. 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),	`LUIXNAM={LUIDXIT|name}`

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

Parameter	Description	Syntax
LUSAVE	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". If LUSAVE is not specified, the default is zero (no user save area is passed).	`LUSAVE={72|size}`

LX1NAME: User Exit 1 Module Name

Parameter	Description	Syntax
LX1NAME	The name of the link user exit 1 module	`LX1NAME={LUEXIT1|name}`

LX2NAME: User Exit 2 Module Name

Parameter	Description	Syntax
LX2NAME	The name of the link user exit 2 module	`LX2NAME={LUEXIT2|name}`

MRO: Multiple Region Option

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

NETOPT: Method Used to Create User ID

Parameter	Description	Syntax
NETOPT	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. 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.	`NETOPT={NO|YES}`

NTGPID: Natural Group ID

Parameter	Description	Syntax
NTGPID	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. There is no default value. If no value is specified, the Adabas internal user ID is built in the conventional manner. 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. 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.	`NTGPID=`*`4-byte-value`*

NUBS: Number of User Blocks Created By CICS Link Routine

Parameter	Description	Syntax
NUBS	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. **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.	`NUBS={100|blocks}`

OPSYS: Operating System

Parameter	Description	Syntax
OPSYS	The operating system in use.	`OPSYS={ZOS|VSE|CMS|BS2}`

PARMTYP: Area for Adabas Parameter List

Parameter	Description	Syntax
PARMTYP	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). 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’). PARMTYP=ALL (the default) uses both the COMMAREA and TWA to pass the Adabas parameters; in this case, the COMMAREA is checked first. 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 in 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.	PARMTYP={ALL|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={LG|*prefix*}

PURGE: Purge Transaction

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

RENT: Reentrant Module Flag

Parameter	Description	Syntax
RENT	Indicates whether the globals module is reentrant.	RENT={NO|YES}

RETRYX: Retry Command Exit Flag

Parameter	Description	Syntax
RETRYX	Indicates whether the retry command exit is active.	RETRYX={NO|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={NO|YES}

RMI: Resource Manager Interface

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

RTXNAME: Command Retry Exit Name

Parameter	Description	Syntax
RTXNAME	The name of the command retry exit module.	RTXNAME={LUEXRTR|*name*}

SAF: Adabas Security Interface Flag

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

SAP: SAP Application Support

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

SAPSTR: SAP ID String

Parameter	Description	Syntax
SAPSTR	The four-byte SAP ID string to use.	SAPSTR={'SAP*'|*string*}

SVCNO: Adabas SVC number

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

TPMON: Operating Environment

Parameter	Description	Syntax
TPMON	The TP monitor operating environment. Valid values should be specified as follows: • 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. **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.**	`TPMON={BAT|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={ADACICT|`*`name`*`}`

UBPLOC: User Block Pool Allocation

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

UBSTIME: User Block Scan Time

Parameter	Description	Syntax
UBSTIME	Specifies the user block (UB) scan time in *fat seconds*. A *fat second* is the interval required to change bit-31 of the doubleword set by an STCK instruction. The default is 1800 seconds. 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. 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. **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.	`UBSTIME={`*`seconds`*`|1800}`

UBTYPE: User Block Type

Parameter	Description	Syntax
UBTYPE	Identifies the kind of user block (UB) storage the Adabas CICS installation program and Adabas task-related user exit (TRUE) should obtain and use. 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. 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. The decision to use UBTYPE=TASK should be based on whether your answers to the following questions are "Yes": 1. Do the majority if 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? UBTYPE=POOL should be used if there are problems with CICS storage fragmentation of when most of the Adabas CICS transactions issues a relatively small number of Adabas calls per CICS task. 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.	`UBTYPE={POOL|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	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. The CICS WAIT EVENTS EXTERNAL (XWAIT=YES) is the recommended interface for CICS/TS 1.1 and above. 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. **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.	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.