Adabas

Installation for z/OS

Version 8.6.2

October 2026

ADABAS & NATURAL

This document applies to Adabas Version 8.6.2 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1971-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software GmbH and/or its subsidiaries is located at https://softwareag.com/licenses.

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software GmbH.

Document ID: ADAMF-OS390-INSTALL-862-20261013

Table of Contents

Preface v
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
2 Supported Environments 5
3 Installation Methods 7
4 Installation Overview 9
Installation Checklist 10
Contents of the Release Tape 11
Migrate an Existing Database 14
Installing the Adabas Release Tape 14
Adabas and Add-on Licenses 15
Disk Space Requirements 16
Initializing the Adabas Communication Environment 18
SVC Integrity Validation 28
Requirements for Cross-Memory Services 29
Device and File Considerations 31
Requirements for Global Resource Serialization 33
Using EXCPVR 34
Creating a Shareable ADARUN 34
Storage Above 16 MB 35
Storage Above 2 GB (64-Bit) 35
Pageable Large Page Support 37
Applying Zaps 39
5 Installing Base Adabas Manually 41
Step 1: Check, Prepare, and Install the Product License File (SMA job number I007) 42
Step 2: Allocate and Format the Adabas Database (SMA Job Number I030) 44
Step 3: Define the Global Database Characteristics (SMA Job Number I030) 44
Step 4: Load the Demonstration Files (SMA Job Number I050) 44
Step 5: Customize and Start the Adabas Nucleus and Test Adabas Communications (SMA Job Number I040) 45
Step 6: Load Adabas Online System Add-On Product, If Appropriate (SMA Job Number I061) 46
6 Installing Adabas with Workflows 47
Before you Install Adabas with z/OSMF Workflows 48
To Install an Adabas Environment with Workflows 49
Variables Reference for Workflows 50
7 Post Install and Verification 51
Step 1: Test Adabas Address Space Communications, If Appropriate 52
Step 2: Terminate the Adabas Nucleus 52
Step 3. Back Up the Database 52

Step 4: Insert the ADARUN Defaults ... 53
Step 5: Install the Required TP Link Routines for Adabas ... 53
8 Installing Adabas Client Link Routines ... 55
Preparing Adabas Link Routines ... 56
Installing Adabas with IMS TM ... 62
General Considerations for Installing Adabas with CICS ... 65
Installing Adabas with CICS ... 66
Installing the CICS High-Performance Stub Routine ... 80
Installing Adabas with Com-plete ... 96
General Considerations for Installing Adabas with Batch/TSO ... 98
Installing Adabas with Batch/TSO ... 99
Establishing Adabas SVC Routing by Adabas Database ID ... 105
Modifying Source Member Defaults (LGBLSET Macro) ... 113
9 Installing Adabas Add-ons (including zIIP) ... 131
License ... 132
Using the Adabas System Coordinator-based Add-ons ... 133
10 Enabling Universal Encoding Support (UES) for Your Adabas Nucleus ... 135
11 Enabling Direct TCP/IP Access (ADATCP) to Your Adabas Nucleus ... 137
12 Installing The AOS Demo Version ... 139
AOS Demo Installation Procedure ... 140
Installing AOS with Natural Security ... 141
Setting the AOS Demo Version Defaults ... 142
13 Installing the Recovery Aid (ADARAI) ... 147
ADARAI Installation Overview ... 148
ADARAI Installation Procedure ... 148
14 Installing The Error Handling And Message Buffering Feature ... 151
15 Adabas Dump Formatting Tool (ADAFDP) ... 153
ADAFDP Function ... 154
ADAFDP Output ... 154
16 Translation Tables ... 161
Adabas EBCDIC to ASCII and ASCII to EBCDIC ... 162
Entire Net-Work EBCDIC to ASCII and ASCII to EBCDIC ... 163
17 Installing Adabas Online Services for z/OSMF Utility Server ... 165
AOZ Server Licensing Requirements ... 166
AOZ Server Installation Steps ... 166
Index ... 169

Preface

This document is intended for those who plan or perform Adabas installation on z/OS systems, and for those who manage or maintain an Adabas database system (such as database administrators and systems programming personnel).

Supported Environments

Installation Methods

Installation Overview

Installing Base Adabas Manually

Installing Adabas with Workflows

Post Install and Verification

Installing Adabas Client Link Routines

Installing Adabas Add-ons (including zIIP)

Enabling Universal Encoding Support (UES) for Your Adabas Nucleus

Enabling Direct TCP/IP Access (ADATCP) to Your Adabas Nucleus

Installing the AOS Demo Version

Installing the Recovery Aid (ADARAI)

Installing The Error Handling and Message Buffering Feature

Adabas Dump Formatting Tool (ADAFDP)

Translation Tables

Installing Adabas Online Services for z/OSMF Utility Server

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

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Supported Environments

Adabas supports a variety of operating environments and can be used in distributed environments. For information on the support platforms for this release of Adabas, read *Supported Platforms*, in the *Adabas Release Notes*.

For specific information regarding Software GmbH product compatibility with IBM platforms and any IBM requirements for Software GmbH products, visit Software GmbH's ***IBM Supported Platforms*** web page.

3 Installation Methods

You can install the product using either a Workflow Installation or a Manual Installation.

Workflow Installation

The Workflow Installation uses a z/OSMF workflow to guide you through the installation process. It provides a structured sequence of tasks, tracks progress, and helps reduce the risk of missing required steps.

Manual Installation

The Manual Installation follows the procedures described in this guide. It is suitable if you prefer a documentation-based installation process or if a z/OSMF workflow is not available in your environment.

Both installation methods achieve the same result. Choose the method that best fits your environment and installation practices.

4 Installation Overview

- Installation Checklist 10
- Contents of the Release Tape 11
- Migrate an Existing Database 14
- Installing the Adabas Release Tape 14
- Adabas and Add-on Licenses 15
- Disk Space Requirements 16
- Initializing the Adabas Communication Environment 18
- SVC Integrity Validation 28
- Requirements for Cross-Memory Services 29
- Device and File Considerations 31
- Requirements for Global Resource Serialization 33
- Using EXCPVR 34
- Creating a Shareable ADARUN 34
- Storage Above 16 MB 35
- Storage Above 2 GB (64-Bit) 35
- Pageable Large Page Support 37
- Applying Zaps 39

Installing Adabas on a z/OS system follows a number of procedures, involves several components, and has certain requirements. The following section provides an overview.

Installation Checklist

The following is an overview of the steps for installing Adabas on a z/OS system.

Important: Be sure that you apply all supplied Adabas 8 maintenance and concatenate Adabas 8 patch-level libraries (L00*n*), as they are delivered to you. This will ensure that your Adabas 8 code remains up-to-date, supporting all Adabas 8 features as they are enhanced and maintained.

Step	Description	Additional Information
	Basic Installation Steps	
1	Allocate DASD space for the Adabas libraries.	The libraries are restored from the installation tape. Refer to the section *Disk Space Requirements for Libraries*.
2	Allocate DASD space for the Adabas database.	For better performance, distribute the database files over multiple devices and channels. Refer to the section *Disk Space Requirements for the Database*.
3	Specify the address space for running the Adabas nucleus.	Refer to the section *Adabas Nucleus Address Space Requirements*.
4	Restore the Adabas libraries from the installation tape.	Use the tape positioning information that accompanies the tape. Refer to the section *Installing the Release Tape*.
5	Install the Adabas SVC temporarily or permanently.	Refer to the section *Initializing the Adabas Communication Environment*.
	Database Installation Steps	Steps 6-15 require changes to the setup definitions as described in section *Installing Base Adabas Manually*.
6	Allocate and format the Adabas database with the ADAFRM utility job.	
7	Define the global database characteristics with the ADADEF utility job.	
8	Load the demonstration files with the ADALODE, ADALODV, ADALODM, and ADALODP jobs.	
9	Prepare and install the product license file.	
10	Customize and start the Adabas nucleus and test the Adabas communications with the appropriate ADANUC job.	

Step	Description	Additional Information
11	If appropriate, test Adabas address space communications by running ADAREP in MULTI mode with the `CPEXLIST` parameter.	
12	If appropriate, load the Adabas Online System (AOS) add-on product into a Natural system file by running the ADAINPL job. Alternatively, install the AOS demo version (see section ***Installing the AOS Demo Version***). **Note:** Demo versions of Adabas Vista (AVI), Adabas Fastpath (AFP), Adabas SAF Security (AAF), and Adabas Transaction Manager (ATM) are automatically installed when you install either the demo or full version of AOS. If you are using the full version of any of the above products, use the SYSPROD Natural system command on each full version product to check if you have installed the latest cumulative INPL update from Empower. Install any missing updates as described in the specific product's documentation.	
13	Terminate the Adabas nucleus with an `ADAEND` operator command using the OS Modify command `F`.	
14	Back up the database by running the ADASAV utility job.	
15	Insert the ADARUN defaults by running the DEFAULTS job.	
	TP Monitor Installation	
16	Install the required TP link routines for Adabas.	See section ***Installing Adabas With TP Monitors***.

Contents of the Release Tape

The following table describes most of the libraries included on the release tape. Once you have unloaded the libraries from the tape, you can change these names as required by your site, but the following lists the names that are delivered when you purchase Adabas for z/OS environments.

Library Name	Description
ACI*vrs*.JOBS	The sample job library for the Adabas CICS link routine. The *vrs* in the library name represents the *version* of the Adabas CICS link routine.
ACI*vrs*.LOAD	The load library for the Adabas CICS link routine. The *vrs* in the library name represents the *version* of the Adabas CICS link routine.
ACI*vrs*.SRCE	The source library for the Adabas CICS link routine. The *vrs* in the library name represents the *version* of the Adabas CICS link routine.
ADA*vrs*.EMPL	The Employees demo file, containing dummy employee data you can use for testing Adabas. The *vrs* in the library name represents the *version* of Adabas.

Library Name	Description
ADA*vrs*.ERRN	Error messages for the Adabas Triggers and Stored Procedures Facility. These messages can be viewed using the Natural SYSERR utility. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.INPL	The code for Adabas Online System, Adabas Caching Facility, Triggers and Stored Procedures Facility, and various add-on demo products. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.JOBS	The sample job library for Adabas. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.LOAD	The load library for Adabas. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.MISC	The Miscellaneous demo file, containing dummy miscellaneous data you can use for testing Adabas. The *vrs* in the library name represents the *version* of Adabas.
ADA*vrs*.PERC	The Personnel demo file, containing uncompressed dummy personnel data you can use for testing Adabas. This demo file includes fields that make use of the extended and expanded features of Adabas 8, include large object (LOB) fields. The *vrs* in the library name represents the *version* of Adabas. **Note:** The Personnel demo file must be installed on a UES-enabled database because it includes wide-character format (W) fields.
ADA*vrs*.SRCE	The source library for Adabas. The *vrs* in the library name represents the *version* of Adabas. For a complete list of the time zones supported by Adabas in any given release, refer to the TZINFO member in this Adabas library.
ADA*vrs*.TZ00	The time zone library for Adabas. The *vrs* in the library name represents the *version* of Adabas. Adabas bases its time zone library on the time zones defined in the public domain **tz database**, also known as the *zoneinfo* or *Olson* database. For a complete list of the time zones supported by Adabas in any given release, refer to the TZINFO member in the Adabas source library (ADA*vrs*.SRCE).
ADA*vrs*.VEHI	The Vehicles demo file, containing dummy vehicle data you can use for testing Adabas. The *vrs* in the library name represents the *version* of Adabas.
AII*vrs*.LOAD	The load library for the Adabas IMS link routine. The *vrs* in the library name represents the *version* of Adabas.
AII*vrs*.SRCE	The source library for the Adabas IMS link routine. The *vrs* in the library name represents the *version* of Adabas.
MLC*vrs*.JOBS	The sample job library for Software GmbH's common mainframe license check software. The *vrs* in the library name represents the *version* of the license check software, which is not necessarily the same as the version of Adabas.
MLC*vrs*.LOAD	The load library for Software GmbH's common mainframe license check software. The *vrs* in the library name represents the *version* of the license check software, which is not necessarily the same as the version of Adabas.
WAL*vrs*.JOBS	The sample job library for Adabas components shared by Adabas and other Software GmbH products, such as Entire Net-Work. The *vrs* in the library name represents the *version* of Adabas.

Library Name	Description
WAL*vrs*.LOAD	The load library for Adabas components shared by Adabas and other Software GmbH products, such as Entire Net-Work. The *vrs* in the library name represents the *version* of Adabas.
WAL*vrs*.SRCE	The source library for Adabas components shared by Adabas and other Software GmbH products, such as Entire Net-Work. The *vrs* in the library name represents the *version* of Adabas.
WCA*vrs*.LOAD	The z/OS load library for Entire Net-Work Administration, used by some of the Adabas add-on products. The *vrs* in the library name represents the version of Entire Net-Work Administration.
WCA*vrs*.SRCE	The z/OS source library for Entire Net-Work Administration, used by some of the Adabas add-on products. The *vrs* in the library name represents the version of Entire Net-Work Administration.

Add-On Product Modules Contained in the ADAvrs.LOAD Library

The following add-on products that were previously distributed on separate libraries are now included in the ADA*vrs*.LOAD library:

Component	Old library	New library	Modules
Adabas Caching Facility (ACF)	ACF*vrs*.LOAD	ADA*vrs*.LOAD	ADACSH ADACSZ ADACS1 ADACS6
Adabas Cluster Services (ALS)	ALS*vrs*.LOAD	ADA*vrs*.LOAD	ADAXCF ADAXEC ADAXEL
Adabas Delta Save Facility (ADE)	ADE*vrs*.LOAD	ADA*vrs*.LOAD	ADADSFN ADADSFR ADADSFS ADAMGS DSFUEX1
Adabas Online System (AOS)	AOS*vrs*.LOAD	ADA*vrs*.LOAD	ADAAOS
Adabas Parallel Services (ASM)	ASM*vrs*.LOAD	ADA*vrs*.LOAD	ADASMC ADASML ADASMM ADAS6C

Modules in components ACF, ALS, ADE, AOS and ASM are now included in the ADA*vrs* library. This means that, when applying zaps to components ACF, ALS, ADE, AOS or ASM on z/OS, the SYSLIB DD in the IMASPZAP job must point to the ADA*vrs* load library. Note that the zap identifiers are unchanged (AH*vrsnnn* for ACF, AP*vrsnnn* for ALS, AL*vrsnnn* for ADE, AS*vrsnnn* for ASM).

For users installing via System Maintenance Aid (SMA), the SMA copy jobs have been updated to take account of the streamlined product delivery. Users installing products via SMA should refer to the *System Maintenance Aid* documentation.

To use the various sub-products which are now delivered with ADA`vrs`, it is necessary for users to install a valid product license for the component. For information on installing product licenses, please refer to ***Adabas and Add-on Licenses*** and the description given in ***Installing Base Adabas Manually***.

Migrate an Existing Database

Use the ADACNV utility to migrate existing databases to new releases of Adabas (SMA job number I021). See *Adabas Utilities* for more information.

Installing the Adabas Release Tape

Copy the data sets from the supplied installation medium to your disk before you perform the individual installation procedure for each component to be installed.

The way you copy the data sets depends on the installation method and the medium used:

- If you use System Maintenance Aid (SMA), refer to the copy job instructions provided in the *System Maintenance Aid* documentation.
- If you are not using SMA and want to copy the data sets from CD-ROM, refer to the README.TXT file on the CD-ROM.
- If you are not using SMA and want to copy the data sets from tape, follow the instructions in this section.

To copy all data sets from tape to disk

1 **Copy Data Set `COPY.JOB` from Tape to Disk**

- Modify the following sample job according to your requirements:

```
//SAGTAPE JOB SAG,CLASS=1,MSGCLASS=X
              //* ---------------------------------
              //COPY EXEC PGM=IEBGENER
              //SYSUT1 DD DSN=COPY.JOB,
              // DISP=(OLD,PASS),
              // UNIT=(CASS,,DEFER),
              // VOL=(,RETAIN,SER=tape-volser),
              // LABEL=(2,SL)
              //SYSUT2 DD DSN=hilev.COPY.JOB,
              // DISP=(NEW,CATLG,DELETE),
              // UNIT=3390,VOL=SER=disk-volser,
              // SPACE=(TRK,(1,1),RLSE),
              // DCB=*.SYSUT1
              //SYSPRINT DD SYSOUT=*
              //SYSIN DD DUMMY
              //
```

where:

tape-volser is the VOLSER of the tape, for example: `T12345`,
hilev is a valid high-level qualifier, and
disk-volser is the VOLSER of the disk.

- Execute the job to copy the data set `COPY.JOB` to your disk.

2 **Modify *hilev*.COPY.JOB on Your Disk**

- Modify *hilev*.`COPY.JOB` according to your requirements:
- Set `EXPDT` to a valid expiration date, for example, `99365`.
- Set `HILEV` to a valid high-level qualifier, for example, `USERLIB`.
- Set `LOCATION` to a storage location, for example, `STORCLAS=ABC` or `UNIT=3390,VOL=SER=USR123`.

3 **Submit COPY.JOB**

Execute *hilev*.`COPY.JOB` to copy single, multiple, or all data sets to your disk.

Adabas and Add-on Licenses

Beginning with Version 8.5.1, the Adabas installation package includes a set of sub-products or add-ons. In order to be able to use the sub-products they must be licensed and the corresponding product license file must be provided with the nucleus or utility execution. The following table describes for a product the member name of the license load module created with LICUTIL and the DD/link name for the license file:

Product	Product Code	Load Module	DD/Link Name
Adabas	ADA	ADALIC	DDLIC
Adabas for zIIP	AZPAD	AZPADLIC	DDLAZPAD
Adabas Replication for ZIIP	AZPRP	AZPRPLIC	DDLAZPRP
Adabas Caching Facility	ACF	ACFLIC	DDLACF
Adabas Delta Save	ADE	ADELIC	DDLADE
Adabas Encryption	AEZ	AEZLIC	DDLAEZ
Adabas Auditing	ALA	ALALIC	DDLALA
Adabas Cluster Services	ALS	ALSLIC	DDLALS
Adabas Parallel Services	ASM	ASMLIC	DDLASM
Adabas Online Services	AOS	AOSLIC	DDLAOS
Adabas Replication Facility	ARF	ARFLIC	DDLARF

Disk Space Requirements

- Disk Space Requirements for Libraries
- Data Sets Required for UES Support
- Disk Space Requirements for ICS Library
- Disk Space Requirements for the Database
- Adabas Nucleus Address Space Requirements

Disk Space Requirements for Libraries

The minimum 3390 disk space requirements for the Adabas libraries are as follows:

Library	3390 Cylinders	3390 Tracks	Directory Blocks
Load	20	300	30
Source	4	60	20
JCL	1	15	20

Note: You can isolate user programs from the Adabas load library by creating a separate load library that contains only those modules needed to execute user programs in multi-user mode and linked with ADAUSER. For this Adabas version, the modules required by user programs are ADAIOR, ADAIOS, ADALNK, ADAMLF, ADAPRF, ADARUN.

Data Sets Required for UES Support

The ICU for Adabas & Natural (ICS) library is required if you intend to enable a database for universal encoding service (UES) support. This library is delivered separately from the product libraries.

For UES support, the following library must be loaded and included in the STEPLIB concatenation, where *nnn* is the load library level and *vrs* is the number of the latest *version* of that code delivered on the tape:

```
ICSvrs.Lnnn
```

For information about setting up connections to UES-enabled databases, see section ***Enabling Universal Encoding Support (UES) for Your Adabas Nucleus***, elsewhere in this guide.

Disk Space Requirements for ICS Library

The minimum disk space requirements on a 3390 disk for the internal product libraries delivered with this version of Adabas are as follows (where *vrs* is the latest *version* of the code delivered on the tape):

Library	3390 Cylinders	3390 Tracks	Directory Blocks
ICS*vrs*.L*nnn*	100	1500	500

Disk Space Requirements for the Database

The actual database space needed by Adabas depends on user requirements. The minimum 3390 disk space requirements for the database are as follows:

Database Component	3390 Cylinders	3390 Tracks
ASSOR1 (Associator)	20	300
DATAR1 (Data Storage)	60	900
WORKR1 (Work space)	15	225
TEMPR1 (temporary work space)	15	225
SORTR1 (sort work space)	15	225

Adabas Nucleus Address Space Requirements

The typical Adabas nucleus requires at least 800-1024 kilobytes to operate. The size of the nucleus address space may need to be larger, depending on the ADARUN parameter settings. Parameter settings are determined by the user.

Initializing the Adabas Communication Environment

This section describes the installation of the Adabas router (ADASVC). The router uses cross-memory services for communication between the Adabas nucleus and the Adabas users.

The Adabas z/OS cross-memory communications service comprises two modules:

- the Adabas router (ADASVC); and
- the Adabas subsystem initialization routine (ADASIR).

ADASIR, executed either during IPL or by the Adabas SVC installation program (ADASIP), initializes the router's operating environment, particularly the ID table.

ADASVC installation can be either temporary or permanent:

- The Adabas SVC can be installed temporarily by executing ADASIP. The SVC is then available only until the next IPL.

  Note: Once installed, the Adabas SVC can be re-installed temporarily using the ADASIP REPLACE option. However, no Adabas nucleus can be active during this procedure.

  Note: It is necessary to cycle CICS after executing ADASIP to initialize the SVC.

- The Adabas SVC is installed permanently using regular operating systems procedures. The SVC then requires an IPL to become active.

Typically, the Adabas SVC is first installed temporarily using ADASIP. This makes Adabas available immediately without the need to wait for an IPL. Meanwhile, preparations are usually made for permanent installation at the next IPL.

- SVC Compatibility Issues Between Adabas Releases
- Authorization Requirements
- Allocating an SVC Table Entry
- Subsystem Name Requirements
- Page-Fixing the Adabas SVC
- Initializing the Adabas SVC
- Router Installation Overview
- Using ADASIP for Temporary Installations

- Using ADASIR

SVC Compatibility Issues Between Adabas Releases

Adabas 8 includes a new Adabas SVC. This SVC is fully backward compatible.

However, you *cannot* use the Adabas SVC from previous Adabas releases with Adabas 8 databases. If you attempt to do this, the Adabas 8 database will not initialize successfully.

Adabas levels are known by their VRS specifications: Version, Release, and SM level. SM level is not a factor in determining compatibility.

With Adabas 8.2 and later versions, the SVC includes performance improvements and improved error recovery routines. Note that the new SVC uses more efficient operating system interfaces, in particular when posting the user at command completion. This shifts work from SRB-mode routines to TCB-mode routines and also between the user's program and the Adabas nucleus. Take this into account when analyzing Adabas 8 SVC performance. With the new SVC, SRB-mode overhead is largely eliminated and TCB-mode overhead is somewhat increased, but the net result is an overall improvement in SVC performance.

Authorization Requirements

The Adabas 8.2 (and later) SVC requires that the Adabas nucleus, as well as other MPM servers (such as Entire Net-Work and the Natural Global Buffer Pool), be authorized to prevent inappropriate use of critical ADASVC functions. This APF authorization prevents unauthorized use of the ADASVC 0-callWe recommend strongly that you run APF-authorized because of the security risks you can incur if you do not. However, upon request, Software GmbH does have a zap you can apply that eliminates this requirement. To determine what this zap number is, review the ZAPOPT member in the Adabas source library for a zap entitled "Remove requirement for APF authorization". If you choose not to have Adabas in an APF-authorized library, modules ADASIP, ADASIR and ADASVC must be copied to an APF-authorized library. Member JCLAPF in the MVSJOBS dataset has sample JCL that invokes z/OS utility IEBCOPY to copy them.

Note: Some add-on products require APF authorization to use restricted z/OS services. APF authorization is still required in these cases.

There are two authorization mechanisms: System Authorization Facility (SAF) and APF authorization. SAF is the z/OS standard interface to security products such as RACF, ACF2, and Top Secret. APF is the z/OS facility that allows programs from designated libraries to access restricted z/OS functions.

The SAF authorization check occurs first. It requests read access to an entity in the FACILITY class or an alternative class specified via `ADASIP` or `ADASIR` parameters when installing the SVC. Refer to the discussion of `ADASIP` and `ADASIR` parameters.

```
ADABAS.SVCsss.IDddddd
```

The *sss* is the Adabas SVC number and *ddddd* is the DBID with leading zeros.

If the nucleus is using Adabas Cluster Services or Adabas Parallel Services, the entity has an additional level:

```
ADABAS.SVCsss.IDddddd.Nucnnnnn
```

The *nnnnn* is the value assigned by the ADARUN NUCID parameter with leading zeros.

The SAF security administrator can assign permissions to entities of this form with a wide range of specificity using a flexible array of patterning and wildcard characters. In general, it should not be necessary to enumerate each possible combination of SVC, DBID, and NUCID.

There are three possible outcomes to the SAF check: permission is explicitly granted, permission is explicitly denied, or the entity may be unknown to SAF. When the entity is unknown, APF authorization is required.

APF Authorization Status	SAF Allow	SAF Deny	Unknown to SAF
APF authorized	Allow	ABEND U494	Allow
Not APF authorized	Allow	ABEND U494	ABEND U494

Message ADAS33 is issued before any ABEND U494.

Note:

APF authorization is still required if your Adabas nucleus configuration needs to use restricted z/OS services. For example, the **EXCPVR option** and some add-on products such as Adabas Cluster Services and Adabas SAF Security (ADASAF) cannot run without APF authorization. In these cases, ADASVC SAF may be used only to deny permission to initialize.

The ADASAF add-on product can also restrict access to a DBID/SVC combination (but not an Adabas Cluster Services or Adabas Parallel Services nucleus ID). ADASAF uses a different resource class and arranges the subfields in resource entities differently. You may choose to use either SAF mechanism, both, or neither. To maintain their existing behavior, current ADASAF users should not create ADABAS.SVC* resource profiles in the FACILITY class and need not change anything in their existing ADASAV configuration.

APF authorization requires that all libraries in the JOBLIB and STEPLIB concatenations have entries in the z/OS APF list. Systems programming staff typically administer the APF list.

Allocating an SVC Table Entry

Regardless of the installation procedure selected, an available SVC table entry must be allocated to the Adabas router (ADASVC). SVC table entries are defined in the member IEASVCxx of SYS1.PARMLIB.

The SVC table entry in the operating system for an ADASVC must contain the following information:

Offset	Label	Description
0	SVCEP	SVC entry point address.
4	SVCATTR1	Must indicate type 2 SVC (flag bit SVCTP2 set—X'80') or type 3 or 4 SVC (flag bits SVCTP34 set—X'C0'): ADASIR changes a type 1, 5, or 6 SVC to type 2. May indicate that APF-authorization is needed for this SVC (flag bit SVCAPF set—X'08'): if set, all targets and users must be APF-authorized.
6	SVCLOCKS	Must contain all zeros. ADASIR sets SVCLOCKS to zeros.

Subsystem Name Requirements

The subsystem name contained in the four-character field SUBSYS at ADASVC offset X'28' (the default is "ADAB") must be the same as that specified in the IEFSSN*xx* member of SYS1.PARMLIB. If the name is not the same, ADASIR ends with an ADAS12 message and condition code 2, and Adabas is not usable.

Page-Fixing the Adabas SVC

If the Adabas SVC is to reside in the fixed LPA, add an entry to an IEAFIX*xx* member of SYS1.PARMLIB.

Initializing the Adabas SVC

The Adabas SVC should be initialized with ADASIP/ADASIR in order to guarantee full functioning of all Adabas nuclei.

Note: The ADASIR routine is prepared for concurrent executions and makes use of PLO (perform locked operation) and CS (compare and swap) instructions when updating shared control blocks. Thus, the BEGINPARALLEL setting is supported (IEFSSN*xx*).

Router Installation Overview

- Temporary Router Installation (SMA Job Number I011)
- Permanent Router Installation (SMA Job Number I010)

Temporary Router Installation (SMA Job Number I011)

Sample jobs may be found in the MVSJOBS dataset from the Adabas installation tape.

to perform temporary router installation:

1. If the Adabas load library is not APF-authorized, customize sample job JCL APF to copy ADASIP, ADASIR and ADASVC into an APF-authorized library as authorized modules.
2. Execute ADASIP to install the SVC.

   Customize and run the job ADASIP to dynamically add the Adabas SVC without an IPL.

Permanent Router Installation (SMA Job Number I010)

Sample jobs may be found in the MVSJOBS dataset from the Adabas installation tape.

to perform permanent router installation:

1. If the Adabas library is not APF-authorized, customize sample job JCLAPF to copy ADASIP to an APF-authorized library.
2. Customize job JCLSVC step COPY to copy ADASVC to SYS1.LPALIB (or another library concatenated with it by SYS1.PARMLIB member LPALST*xx*). Standard practice is to rename it according to z/OS SVC naming rules. For example, type 3/4 SVCs are named IGC00*nnn*, where *nnn* is the SVC number expressed as signed decimal digits. A signed decimal is a number ends in either of the following ways:

   - When the last digit of the SVC routine's load module name is a number from 1 - 9, specify a name that ends with the EBCDIC character (A-I) that corresponds with the last digit. For example, the name for type 3/4 SVC 249 is IGC0024I.
   - When the last digit of the SVC routine's load module name is zero, specify for the last character of the name the display representation of hexadecimal C0; in EBCDIC this is the left brace ({) character.

   z/OS allows the type 3/4 naming convention to be overridden by the entry in SYS1.PARMLIB member IEASVC*xx*. The load module name specified in the IEASVC*xx* entry takes precedence. For example, if you have only one instance of ADASVC in the LPA libraries you may specify that name in IEASVC*xx* and leave the actual load module name unchanged.

3 If you wish to associate a subsystem name other than the default ADAB with ADASVC, customize sample job JCLSVC step SUBSYS to invoke IBM utility IMASPZAP to set the desired 4-character subsystem name.

4 Add a subsystem entry to SYS1.PARMLIB member IEFSSN*xx* for the subsystem name associated with the SVC. Customize and run the job JCLUPDT to add a new subsystem definition entry in the correct format.

   Or:

   Make the update with an on-line editor.

5 Customize sample job JCLSIR to copy ADASIR to SYS1.LINKLIB or a library concatenated with it by SYS1.PARMLIB member LKNLST*xx*.

   Note: ADASIR is not reentrant, and therefore should not be linked into SYS1.LPALIB or a library concatenated with it by SYS1.PARMLIB member LKNLST*xx*.

6 IPL z/OS with the CLPA option to install and initialize the Adabas communication environment.

Using ADASIP for Temporary Installations

- ADASIP Functions
- ADASIP Parameters
- Executing ADASIP

ADASIP Functions

ADASIP performs the following functions:

- acquires memory in the specified CSA subpool for the Adabas SVC and a subsystem communication vector table (SSCT)
- loads the Adabas SVC into the acquired CSA space
- modifies the subsystem name at ADASVC offset +x'28'
- modifies the SVC table entry as required by the Adabas SVC
- optionally deletes an SSCT for the same subsystem name from the SSCT chain
- adds the new SSCT to the SSCT chain
- invokes the ADASIR program
- releases CSA acquired by a previously installed SVC

If any error is detected, ADASIP backs out all completed activities and terminates operation with a user abend specifying the error.

When reinstalling an instance of ADASVC using an SVC number that is currently being used by ADASVC, the subsystem name must be the same as the one currently being used. This helps avoid a configuration that may not function correctly. For more information, read ***SVC Integrity Validation***, elsewhere in this guide.

A Version 8 ADASIP/ADASIR can be used to install a Version 7 Adabas SVC, but the Version 8 SVC requires the Version 8 ADASIP/ADASIR. we recommend using Version 8 to install all supported releases of the SVC. If an earlier version of ADASIP/ADASIR is used to replace an SVC installed with a later version, some areas of common storage may not be released.

If you choose not to have Adabas in an APF-authorized library, modules ADASIP, ADASIR and ADASVC must be copied to an APF-authorized library. Member JCLAPF in the MVSJOBS dataset has sample JCL that invokes IEBCOPY to copy them.

ADASIP Parameters

ADASIP parameters have the following syntax:

```
CONSNAME=c,IDTSPL=i,LEAVE=l,NRIDTES=n,REPLACE=r,SUBSYS=su,
SVCNR=svcn,SVCSPL=svcs,RCLASS=class,FORCE=force
```

—where

c	is the console name to which operator messages are written. If omitted, messages are issued using ROUTCDE=2, Master Console Information.
i	is the ID table subpool: see the ADASIR `IDTSPL` parameter for details.
l	indicates whether ADASIR should display message ADAS11 or ADAS12 on the operator console: see the ADASIR `LEAVE` parameter for details.
n	is the number of ID table entries: see the ADASIR `NRIDTES` parameter for details.
r	indicates whether or not an existing SSCT for the same subsystem name is to be replaced. Y for yes or N for no (N is the default). Use this option to replace any type of Adabas SVC (for example, when installing a new SVC version).
su	is the subsystem name. This parameter is required. . Each instance of the Adabas SVC must have a unique subsystem name.
svcn	is the Adabas SVC number: see the ADASIR `SVCNR` parameter for details.
svcs	is the Adabas SVC and SSCT subpool: 228 for fixed CSA or 241 for pageable CSA (default: 241).
class	is the System Authorizatin Facility (SAF) class name used to test authorization to start a target (default FACILITY).
force	allows certain possible errors to be detected while validating parameters to be overridden. Use `Y` for overriding or `N` for no overriding (`N` is the default).

The following are valid ADASIP parameter abbreviations:

Parameter	Abbreviation
CONSNAME=	C=
IDTSPL=	I=
LEAVE=	L=
NRIDTES=	N=
REPLACE=	R=
SUBSYS=	SU=
SVCNR=	SVCN=
SVCSPL=	SVCS=
RCLASS=	RC=
FORCE=	F=

All parameters are optional except SUBSYS and SVCNR. If specified, the parameters IDTSPL,LEAVE, NRIDTES, SUBSYS, SVCNR, and RCLASS

are passed to ADASIR without being verified.

Executing ADASIP

JCL similar to the following should be used to execute ADASIP:

```
// EXEC PGM=ADASIP,PARM=parameters
//STEPLIB DD ...
//SVCLIB DD ...
//SIRLIB DD ...
```

There is a sample job in member JCLSIP of the MVSJOBS dataset.

The data set defined by the STEPLIB DD statement must be an APF-authorized library containing the APF-authorized program ADASIP. Since ADASIP is neither reentrant nor refreshable, the data set cannot be SYS1.LPALIB.

The data set defined by the SVCLIB DD statement must be an APF-authorized library containing the Adabas SVC with either the name or alias ADASVC.

The data set defined by the SIRLIB DD statement must contain the ADASIR program. Since ADASIR is neither reentrant nor refreshable, the data set may not be SYS1.LPALIB.

ADASIP terminates with a U0481 abend if the parameter input is incorrectly specified.

The ADASIP parameters may not fit on a single JCL statement and may be broken up into segments using continuation JCL statements. The IBM job control convention for continuing the PARM parameter is:

```
// EXEC PGM=ADASIP,PARM=('parameters ....',
// 'parameters')
```

Note these restrictions:

- Enclose the entire PARM string in parentheses.
- Place each segment of the PARM string in single quotes.
- Break the PARM string before a comma that follows a parameter.
- A comma is required after the end-quote on a line that is to be continued.
- Start the continuation line within columns 4-16.
- A comma is not permitted between the last parameter and the end-quote on the line to be continued because JCL automatically inserts a comma between parameters when concatenating continuation strings:

  ```
  // ...PARM=('CONSID=3',
  // 'SUBSYS=ADAB',
  // 'SVCNR=249')
  ```

  —results in an equivalent line of

  ```
  CONSID=3,SUBSYS=ADAB,SVCNR=249
  ```

Using ADASIR

- ADASIR Functions
- ADASIR Parameters
- Executing ADASIR

ADASIR Functions

The ADASIR program is invoked

- by the ADASIP program to install the Adabas SVC temporarily, or
- by z/OS to install the Adabas SVC permanently.

ADASIR receives control during either master scheduler initialization or ADASIP execution. The operator is prompted for any value that has been incorrectly zapped or assembled (refer to the *Adabas Messages and Codes* for specific message descriptions). If an error is found during the processing of parameters specified in the IEFSSN*xx* member or passed by ADASIP, the operator is prompted for all of the values.

If the SVC table entry is incorrect, ADASIR prompts the operator for permission to change the entry (if `SVCTAB=P`, the default, is specified). If any errors are detected, they must be corrected and either another IPL must be done or ADASIP must be rerun before the Adabas SVC can be used.

ADASIR Parameters

ADASIR parameters have the following syntax:

```
IDTSPL=i,LEAVE=l,NRIDTES=n,SVCNR=svcn,SVCTAB=svct,RCLASS=class
```

Variable	Description
i	The ID table subpool: 228 for fixed CSA or 241 (the default) for pageable CSA.
l	Indicates whether message ADAS11 or ADAS12 is to be displayed on the operator console: Y for yes or N (the default) for no.
n	The ID table entry count, which can range from 1 to a maximum specified at offset X'146' in the CSECT IEAVESVT of the z/OS nucleus (see section ***Requirements for Cross-Memory Services***).
svcn	The Adabas SVC number (200-255).
svct	Indicates whether or not the operator should be prompted for permission to update the SVC table entry. Enter P (the default) to receive a prompt, or N for no prompt. P is recommended if a possibility exists that the SVC table entry will not be what ADASIR expects.
class	is the System Authorizatin Facility (SAF) class name used to test authorization to start a target (default FACILITY).

The following are valid abbreviations for ADASIR parameters:

Parameter	Abbreviation
IDTSPL=	I=
LEAVE=	L=
NRIDTES=	N=
SVCNR=	SVCN=
SVCTAB=	SVCT=
RCLASS=	RC=

Executing ADASIR

Customize sample job JCLSIR to copy ADASIR to SYS1.LINKLIB or a library concatenated with it by SYS1.PARMLIB member LKNLST*xx*.

Note: ADASIR is not reentrant, and therefore should not be placed into SYS1.LPALIB or a library concatenated with it by SYS1.PARMLIB member LKNLST*xx*.

To prepare for permanent SVC installation, an entry must be made in either a new or existing member having the name IEFSSN*xx* in SYS1.PARMLIB. This entry is an 80-character record with the following format:

```
SUBSYS SUBNAME(cccc) CONSNAME(consname) INITRTN(ADASIR) ↩
INITPARM('parameters') comments
```

—where

cccc	The 1- to 4-character subsystem name. This name and the name specified in the Adabas SVC at offset X'28' must be the same. The name provided in the SVC is ADAB; any other name must first be zapped into the SVC before being specified for *cccc*.
consname	The name of the console to which ADASIR will direct any messages. If omitted messages will be issued with ROUTCDE=2, Master Console Information.
'parameters'	ADASIR parameters. If there is more than one parameter, values must be enclosed in single quotation marks and a comma placed between the parameters.
comments	Comments are optional and must be preceded by at least one space.

If the subsystem name does not match, ADASIR abends with an ADAS12 message and condition code 2; the Adabas z/OS communication environment is not initialized. Re-IPL z/OS, specifying `SSN=xx` if necessary. If this is the first IPL with a type 3 or 4 Adabas SVC, specify CLPA as one of the `SET` parameters.

If an error is encountered while processing any of the parameters obtained from the IEFSSN*xx* member or passed from ADASIP (message ADAS05), the operator is prompted to reenter all of the parameters. If the SVC table entry is not correct (message ADAS09) then, depending on the value of the `SVCTAB` parameter, either the operator is prompted (message ADAS10) for permission to change the `SVCTAB` parameter, or it is simply changed (message ADAS15).

A Version 8 ADASIR can be used to install a Version 7 Adabas SVC, but the Version 8 SVC requires the Version 8 ADASIR. We recommend using Version 8 to install all supported releases of the SVC.

SVC Integrity Validation

In the past, the presence of multiple SVCs with the same subsystem ID has resulted in a single ID table being used by different SVCs. This has caused problems, some of them , such as abnormal nucleus termination or corruption of the database.

To eliminate this exposure, ADASVC checks to ensure that the SVC accessing the ID table is the same as the one that was used by ADASIP/ADASIR to initialize the table. If the SVCs are not the same, the IDT is not accessed.

The lack of a usable IDT may occur during Adabas initialization, during the first Adabas call from a user program, or when the ID table is queried by another server such as Entire Net-Work.

ADASIP has been enhanced to prevent this from arising. If you attempt to install an instance of ADASVC using an SVC number that is presently associated with another subsystem name, message ADAS16 is issued and ADASIP terminates with user abend 435.

When installing or reinstalling ADASVC, ADASIP will also detect if there is an Adabas SVC already installed as the specified SVC number that specifies a different subsystem than specified by the SUBSYS parameter. This causes message ADAS16 to be issued and ADASIP terminates with user abend 436. This abend can be overridden by specifying the `FORCE=Y` parameter.

If the specified SVC exists and has an IDT where there are active targets (as indicated by an active IDTE), message ADAS17 is issued for each such target and ADASIP terminates with user abend 437. This abend can be overridden by specifying the `FORCE=Y` parameter but doing so may cause the targets to abend when they try to access the IDT. Using the `FORCE=Y` parameter should only be necessary in rare conditions when a target failed in such a way that the IDT could not be cleaned up.

Requirements for Cross-Memory Services

Due to the implementation of cross-memory services in z/OS, the following points should be noted when running an Adabas nucleus in MULTI mode:

- a maximum of one step of a job can establish the cross-memory environment. This means that a job can include at most one step that is a target (for example, an Adabas nucleus).
- cross-memory accesses may not be made to a swapped-out address space. Therefore, the address space of an Adabas nucleus is set to “nonswappable” for the duration of the nucleus session. This can increase the installation’s real storage requirements. This behavior is documented in the IBM manual *Extended Addressability Guide*, chapter Synchronous Cross-Memory Communication.
- when a nucleus with an active cross-memory environment terminates either normally or abnormally, the entire address space including any initiator is also terminated.

The address space ID (ASID) representing this address space is not reassigned until the next IPL. Therefore, you should choose a sufficiently high value for the `MAXUSERS` parameter in the active IEASYS*xx* member of SYS1.PARMLIB or—if your system supports it—the `RSVNONR` parameter in the same member can be adjusted accordingly. Also, the Adabas nucleus should not be stopped and started without good reason. This is described in the manuals referred to in the topics *Recovery Considerations* and *Resource Management*. Additional information can be found in IBM APARs OZ61154, OZ61741, and OZ67637.

To reduce the possibility that the system will run out of ASIDs for assignment to new address spaces, start ADABAS with REUSASID=YES:

```
/S ADAxxxx,REUSASID=YES
```

This is described in the z/OS *V1R12.0 MVS Programming Extended Addressability Guide SA22-7614-07*.

To make its services available to all address spaces in the system, the Adabas nucleus must obtain a system linkage index (LX) from z/OS. The LX is a reserved slot in the linkage space of all address spaces, and permits system-wide linkage to all address spaces.

If your hardware supports the Address Space and LX Reuse Facility (ALRF), the version 8 ADASVC will make use of reusable system LXs. Otherwise, a non-reusable system LX will be used as in previous releases. Reusable system LXs resolve the constraints imposed by non-reusable LXs. The remainder of this section discusses these constraints.

The number of non-reusable LXs set aside by z/OS for system use is rather small (usually 165 out of a possible 2048).

Because of the way z/OS use cross-memory services, non-reusable system LXs obtained by Adabas cannot be returned to z/OS until the next IPL. However, the system that owns the LXs can reuse them, even for a different address space. Adabas makes use of this feature by identifying used LXs in a table in CSA, where they are available to future nuclei.

The number of non-reusable system LXs can be specified in the member IEASYSxx contained in SYS1.PARMLIB, using the `NSYSLX` parameter. If you change this value, you must perform an IPL to make the change effective.

To determine an appropriate `NSYSLX` value, consider the following points:

- some LXs are probably already being used by other system functions. Therefore, the chances of creating an LX shortage for other users is small.
- Adabas requires one system LX for each Adabas nucleus (or any other target) that will be active concurrently. A value of decimal 64 would allow concurrent execution of up to 64 Adabas nuclei or other targets with little chance of restricting other components using LXs.
- Entire Net-Work Version 5 uses only one LX and one ID table entry, regardless of how many remote databases it must represent. This is unlike the pseudo-MPM concept of earlier Entire Net-Work versions.
- whenever ADASIP is executed with the REPLACE option, all LXs saved in the current ID table are lost until the next IPL.

Likewise, if a session ends either normally with the `FORCE` operator command or abnormally during ESTAE processing (for example, by an S222 operator cancel or by a S722 spool limit exceeded abend during a snap dump), the LX also cannot be recovered until the next IPL.

Any commands sent to these targets receive an S0D6 abend. Any attempt to restart the nucleus results in an ADAM98 message DUP ID (LOCAL), followed by an abend. To resolve both of these problems, restart the nucleus with the ADARUN `FORCE=YES` and `IGNDIB=YES` parameters.

The first target that tries to obtain a system LX when none is available ends with an S053 abend code and reason code 0112. No additional targets can be started until the next IPL.

Device and File Considerations

This section provides information on device and system file topics.

- Adabas File Characteristics
- DFSMS-Managed Files
- Large Format File Support
- Supported Adabas Device Types
- Adding or Updating Devices
- Enhanced Backup and Restore Performance in Tape Sequential Files

Adabas File Characteristics

Adabas container data sets (ASSO, DATA, WORK, TEMP, SORT, PLOG, CLOG, RLOG, ALOG, or DSIM) consist of fixed-length records (RECFM=F), with a data set organization which is compatible with sequential data sets (DSORG=PS).

Both basic format and large format data sets are supported for Adabas containers.

Extended format data sets (also known as striped data sets) are not supported for Adabas containers. If an attempt is made to open an Adabas container using an extended format data set, an error message will be issued and the job will be terminated.

DFSMS-Managed Files

Adabas files can be managed by DFSMS, and placed on disk devices belonging to a DFSMS Storage Group. This allows Adabas to take advantage of the storage management features of DFSMS, for example allowing DFSMS to administer the space requirements of data sets when they are allocated or extended.

The procedure for extending Adabas files differs according to whether the file is DFSMS-managed or not. See *DBA Tasks > Database Space Management > Increasing Associator and Data Storage Space* for more details. See also the descriptions of ADABAS INCREASE and ADADBS ONLINCREASE in the Utilities manual.

If Adabas container files are encrypted using the add-on Adabas Encryption product (product code AEZ), DFSMS management is mandatory. Otherwise, it is optional.

Large Format File Support

Adabas supports large format files, a feature introduced with IBM z/OS 1.7.

Prior to large format file support, the extents of an Adabas container (ASSO, DATA, WORK, TEMP, SORT, PLOG, CLOG, RLOG, ALOG, or DSIM data sets) were only able to occupy 65,535 tracks per volume. With large format file support, the data set extents can now occupy an entire volume.

As before, each data set can have up to 16 extents on a volume and can span up to 59 volumes. For ASSO and DATA, up to 99 separate data sets are supported.

In addition to supporting large format files for container data sets, Adabas sequential data sets (typically those read or written by the Adabas utilities) can also take advantage of IBM large format file features.

When allocating a new Adabas data set (container or sequential) where the space on any single volume can exceed 65,535 tracks, the DSNTYPE=LARGE parameter needs to be specified in the JCL. If the data set is eligible to use the cylinder-managed space on an EAV disk device, EATTR=OPT needs to be specified in the JCL, in addition to DSNTYPE=LARGE. Once a large format file is allocated, the DSNTYPE=LARGE and EATTR=OPT paramaters are no longer necessary in JCL for the file.

For further information about the DSNTYPE JCL parameter, as well as for details about the IBM PARMLIB parameter BLOCKTOKENSIZE(REQUIRE) in member IGDSMS*xx*, refer to the IBM z/OS documentation provided in section *Processing Large Format Data Sets* in the *DFSMS: Using Data Sets* guide.

Supported Adabas Device Types

The standard characteristics of the device types supported via the ADARUN DEVICE parameter in z/OS environments are summarized in the following table. Adabas block sizes and blocks per track are provided for each Adabas component for each device type.

Device	Trks/Cyl	ASSO	DATA	WORK	PLOG/RLOG	CLOG/ALOG	TEMP/SORT/DSIM	Notes
3390	15	2544:18	5064:10	5724:9	5724:9	5064:10	8904:6	1
8390	15	3440:14	6518:8	10706:5	10706:5	8904:6	8904:6	1
8391	15	4136:12	10796:5	13682:4	13682:4	8904:6	18452:3	1
8392	15	4092:12	12796:4	18452:3	18452:3	18452:3	18452:3	1
8393	15	4092:12	27644:2	27990:2	27990:2	27990:2	27990:2	1
3380	15	2004:19	4820:9	5492:8	5492:8	4820:9	7476:6	2
8380	15	3476:12	6356:7	9076:5	9076:5	9076:5	9076:5	2
8381	15	3476:12	9076:5	11476:4	11476:4	9076:5	9076:5	2
8385	15	4092:10	23292:2	23468:2	23468:2	23468:2	23468:2	2

Notes:

1. The 3390 and 839*n* device types are to be used for all disks emulating 3390 devices. Device types 839*n* provide larger block sizes than the default 3390.
2. The 3380 and 838*n* device types continue to be supported for 3380 devices. Device types 838*n* provide larger block sizes than 3380.

Adding or Updating Devices

Any users requiring to add a new Adabas device type, or else update an existing Adabas device type, should contact their our support representative for assistance.

Enhanced Backup and Restore Performance in Tape Sequential Files

Adabas exploits IBM's large block (more than 32,760 bytes) support for sequential access methods BSAM and QSAM under z/OS version 2 release 10 and above. ADAIOR supports tape drives with a block size of up to 256K for 3590 devices and 64K for 3490/3490E devices.

This support can provide performance benefits for any utility writing to tape (for example, ADASAV).

Caution: If you choose to write tape files with large block sizes (for example, for database backups), these files will not be transportable to systems where support for large blocks is not available. This might include a site being used as a backup facility for disaster recovery.

Requirements for Global Resource Serialization

Adabas uses Global Resource Serialization (GRS) to synchronize the execution of Adabas nuclei and utilities at certain points in their processing. It is vital that GRS be set up correctly in the system so that GRS requests by Adabas will be effective.

When setting up GRS, consider the following:

- Adabas uses the GRS macros ENQ and DEQ with systems-wide scope (`SCOPE=SYSTEMS`) and major name 'ADABAS' (QNAME).
- if the database resides on disks that are shared between multiple images of the operating system (multiple LPARs or machines) and Adabas nuclei or utilities may be run against the database from several of these images, make sure that GRS is installed in a way that systems-wide ENQ requests are effective on all of these system images.

Using EXCPVR

Adabas performance and system throughput are improved when you use EXCPVR rather than EXCP, due to a reduction in channel program translation overhead.

For Adabas to invoke EXCPVR, the nucleus must be running with APF authorization. All Adabas modules must be in APF-authorized libraries, and all libraries in the JOBLIB or STEPLIB concatenations must be APF-authorized.

Running the Adabas nucleus and utilities APF-authorized is now a standard. For more information, read ***APF Authorization Requirement***, elsewhere in this guide.

Prior versions of Adabas tied the use of EXCP or EXCPVR for database I/Os to the APF-authorization of the load library. EXCP was always used when running non-APF-authorized; EXCPVR was always used when running APF-authorized. If you wanted to use EXCP when running APF-authorized, you were required to apply special A$- or AY- zaps.

Adabas 8 introduces a new ADARUN parameter, `EXCPVR`, available in z/OS environments. Using this parameter, you can specify whether EXCP or EXCPVR is used when running APF-authorized. If the `EXCPVR` parameter is set to "YES", EXCPVR is used; if it is set to "NO", EXCP is used. If the `EXCPVR` parameter is set to "YES" when running from a non-APF-authorized load library, the `EXCPVR` parameter is ignored. For complete information on the use of this parameter, read *EXCPVR : Control EXCP or EXCPVR Use*, in *Adabas Operations Manual*.

In addition to the `EXCPVR` parameter, another ADARUN parameter, `PGFIX`, affects EXCPVR use. When EXCPVR is used on z/OS systems, the `PGFIX` parameter controls whether pages containing I/O control blocks are released after I/O processing is completed or after the job has ended. This parameter is valid only for z/OS users of EXCPVR. When the `PGFIX` parameter is set to NO, pages containing the I/O control blocks are fixed only for the duration of the I/O processing; when it is set to YES, pages containing the I/O control blocks are fixed for the duration of the job. For complete information on the use of this parameter, read *PGFIX: EXCPVR Page Fixing Control*, in *Adabas Operations Manual*.

Creating a Shareable ADARUN

The ADARUN module delivered in the Adabas load library is not reusable. If you need a shareable ADARUN, you will need to relink it with the `REUS=YES` link-edit attribute.

Linking ADARUN with the reusable option permits several programs running in the same address space to share the same ADARUN and ultimately, the same copy of ADALNK. This is important when it is necessary to have only one Adabas user ID for the different programs, and is also needed if single copies of ADALNK user exits are required.

To create a shareable ADARUN, use the sample job JCLLINRR in the JOBS library to relink it with the `REUS` attribute.

If both nonreusable and reusable versions of ADARUN are required, they must be located in different load libraries since both must be loadable using the name ADARUN.

Storage Above 16 MB

Adabas can acquire a number of its required areas, including buffer space, above the 16-MB addressing limit, allowing Adabas to increase the buffer pool size.

To reverse the space allocation to be below the 16-MB limit, set the `AMODE` value in the MODE statement in the example below to `AMODE(24)`:

```
//LINKRUN EXEC PGM=IEWL,PARM='REUS'
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//ADALIB DD DSN=user.loadlib,DISP=SHR
//SYSLMOD DD DSN=user.loadlib,DISP=SHR <=APF-authorized library
//SYSLIN DD *
MODE AMODE(24),RMODE(24)
SETCODE AC(1)
INCLUDE ADALIB(ADARUN)
NAME ADARUN(R)
```

In addition, Adabas must be run with a sufficient REGION specification, either on the JOB or EXEC statement or as an installation default. For example:

```
//BIG JOB ...,REGION=30M,...
```

Storage Above 2 GB (64-Bit)

- Real Storage

- Virtual Storage

Real Storage

Adabas can exploit storage occupying real pages above the 2-gigabyte line. This capability allows Adabas I/Os to use 64-bit real addresses.

Support for 64-bit real storage is available whether you are running APF-authorized (using EXCPVR) or not (using EXCP). The run mode is indicated in the ADAI65 message:

```
ADAI65 EXCPVR IS {BEING | NOT BEING} USED FOR THIS RUN IN ESA64 MODE
```

Support for 64-bit real storage requires either

- z/OS R10 in `ARCHLEVEL=2` (that is, z/architecture mode); or
- z/OS 1.2 or above

on a processor of the IBM 2064 family with an LPAR greater than 2 gigabytes for real storage allocation.

Virtual Storage

IBM supports 64-bit virtual storage only for z/OS 1.2 or above.

Software GmbH provides support for IBM's 64-bit virtual storage in Adabas. The ADARUN V64BIT parameter allows you to indicate whether the Adabas nucleus should use virtual storage above the 2 gigabyte bar. If your z/OS operating system supports 64-bit virtual storage, you can request that the Adabas nucleus use it also. In addition, the ADARUN LARGEPAGE parameter allows you to indicate whether the Adabas nucleus should use large pages (1 MB pages of real storage above the 2 gigabyte bar). If your z/OS operating system supports 64-bit virtual storage and large pages, you can request that the Adabas nucleus use large pages.

we also provides support for 64-bit virtual storage in the product Adabas Caching Facility (ACF). contact your nearest representative for more information.

A demo of Adabas Caching Facility is delivered in the ADA*vrs*.ALLINPL file (where *vrs* is the number of the latest Adabas *version* delivered on the tape).

Pageable Large Page Support

Adabas for Mainframes supports pageable large pages.

Pageable large page support allows storage to be backed by 1M large pages without the configuration restrictions of fixed large page support (e.g. by reserving space via the LFAREA parameter in SYS1.PARMLIB member IEASYSxx).

To check the availability of pageable large pages, refer to the IBM documentation for z/OS and for the features supported by your CPU model.

Additional options to the ADARUN LARGEPAGE parameter are defined, allowing you to indicate whether your Adabas nucleus should attempt to make use of pageable large pages. For more information, see *LARGEPAGE : Large Page Support Option*.

Note that the implementation of pageable large page support by IBM indicates a preference for the storage to be backed by pageable large pages. If no pageable large pages are available, or if certain operations (e.g. page-fixing) are performed against storage backed by pageable large page, storage backed by 1M large page can get demoted to being backed by pageable 4K pages during the run time of the job. Once storage has been demoted to being backed by 4K pages, it will continue to be backed by 4K pages for the remaining duration of the job. The success or otherwise of the request to allocate storage backed by large pages is indicated via the ADAI39 and ADAI44 messages described in the Messages manual. If an area of storage initially backed by large pages is later demoted to 4K pages, there is no indication of this within the Adabas job.

An attempt to allocate storage using pageable large pages will occur in the following situations:

- In an Adabas nucleus running with LARGEPAGE=P64 and V64BIT=YES, an attempt to allocate storage below the 2G bar backed by pageable large pages will be made for the large buffer area associated with various ADARUN parameters, including LBP, LFP, and LWP. The flush I/O pool (ADARUN parameter LFIOP) will be allocated above the 2G bar in storage backed by 4K pages. No attempt is made to allocate flush I/O pool storage above the 2G bar using 1M pageable large pages, as this area is used for I/O and will therefore get demoted to being backed by 4K pages.
- In an Adabas nucleus running with LARGEPAGE=P64 and V64BIT=NO, an attempt to allocate storage below the 2G bar backed by pageable large pages will be made for the large buffer area associated with various ADARUN parameters, including LBP, LFP, LWP, and LFIOP. Users should note that potentially large parts of this buffer area (those used for I/O) are subject to being demoted to being backed by 4K pages. For these areas, any potential performance benefit in using 1M pageable large pages will be lost for the duration of the job.
- In an Adabas Parallel Services environment, specifying CLUCACHETYPE=P64 will attempt to allocate storage above the 2G bar backed by pageable large pages for the global cache area (ADARUN parameter CLUCACHESIZE).

- In an Adabas Parallel Services environment, specifying LARGEPAGE=P64 and V64BIT=YES will attempt to allocate storage above the 2G bar backed by pageable large pages for the cast-out directory buffer (size derived internally and not associated with any single ADARUN parameter).
- With Event Replicator V4.1 SP1 or later, in an Event Replication Server, specifying LARGEPAGE=P64 and V64BIT=YES will attempt to allocate storage above the 2G bar backed by pageable large pages for the replication pool (ADARUN parameter LRPL).
- With versions of Event Replicator prior to V4.1 SP1, in an Event Replication Server, specifying LARGEPAGE=P64 and V64BIT=YES will attempt to allocate storage below the 2G bar backed by pageable large pages for the replication pool (ADARUN parameter LRPL).
- For all supported versions of Event Replicator, in an Event Replication Server, specifying LARGEPAGE=P64 and V64BIT=NO will attempt to allocate storage below the 2G bar backed by pageable large pages for the replication pool (ADARUN parameter LRPL).
- For all supported versions of Event Replicator, an Adabas nucleus running with REPLICATION=YES, specifying LARGEPAGE=P64 and V64BIT=YES will attempt to allocate storage above the 2G bar backed by pageable large pages for the replication pool (ADARUN parameter LRPL).
- For all supported versions of Event Replicator, an Adabas nucleus running with REPLICATION=YES, specifying LARGEPAGE=P64 and V64BIT=NO will attempt to allocate storage below the 2G bar backed by pageable large pages for the replication pool (ADARUN parameter LRPL).
- In an Adabas nucleus running with AUDITING=YES or in an Auditing Server, specifying LARGEPAGE=P64 and V64BIT=YES will attempt to allocate storage above the 2G bar backed by pageable large pages for the auditing pool (ADARUN parameter LAP, described in the *Adabas Auditing* documentation).
- In an Adabas nucleus running with AUDITING=YES or in an Auditing Server, specifying LARGEPAGE=P64 and V64BIT=NO will attempt to allocate storage below the 2G bar backed by pageable large pages for the auditing pool (ADARUN parameter LAP, described in the *Adabas Auditing* documentation).
- In an Adabas nucleus running with the Adabas Caching Facility active, allocation of the various storage areas using pageable large pages will be attempted according to extensions to the associated Adabas Caching Facility parameters. These parameters are described in the ADARUN parameters section of the *Operations* manual.

Applying Zaps

Use the z/OS AMASPZAP utility to apply zaps in the respective operating system; this method verifies (VER) and replaces (REP) data. The following sample JCL executes AMASPZAP:

```
//ADAZAP JOB
//STEP1 EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=X
//SYSLIB DD DSN=ADABAS.ADAvrs.LOAD,DISP=SHR
//SYSIN DD *
(zap control statements)
/*
//
```

—where the following are examples of zap control statements:

```
NAME membername csectname
VER displacement data
REP displacement data
IDRDATA (up to eight bytes of user data)
* (comment)
```

Note: In VER and REP statements, spaces must be used to separate command, displacement, and data. Commas are acceptable data separators; however, commas with spaces or spaces alone are not, and may cause errors.

5 Installing Base Adabas Manually

- Step 1: Check, Prepare, and Install the Product License File (SMA job number I007) 42
- Step 2: Allocate and Format the Adabas Database (SMA Job Number I030) 44
- Step 3: Define the Global Database Characteristics (SMA Job Number I030) 44
- Step 4: Load the Demonstration Files (SMA Job Number I050) 44
- Step 5: Customize and Start the Adabas Nucleus and Test Adabas Communications (SMA Job Number I040) 45
- Step 6: Load Adabas Online System Add-On Product, If Appropriate (SMA Job Number I061) 46

This chapter describes the installation of Adabas on z/OS systems.

Step 1: Check, Prepare, and Install the Product License File (SMA job number I007)

You must install a valid license file on all mainframe platforms in which your Software GmbH mainframe product is installed. The license file is provided as an XML document (encoding is US-ASCII) and must remain in that format -- even on the mainframe. It must not be modified. Any modification of the license file will invalidate the digital signature and the license check will fail. In the event of a check failure, please contact your technical support representative.

For a list of license file names, load modules and DD/Link names, refer to ***Adabas and Add-on Licenses***, elsewhere in this guide.

Note: Thirty days before the license expires, license check failure messages are produced. Your software product will still function, but these messages warn you that it is time to obtain a new license.

In this step, you will prepare the license file (obtain it from e-mail or the installation tape and store it on your z/OS system) and then install it:

Preparing the Product License File

The product license file is supplied on the individual customer installation tape or separately via an e-mail attachment. Before you can install the license, you must transfer it from e-mail or the installation tape and store it on a z/OS system. This section describes how to do this for a license distributed either by e-mail or on the installation tape.

To prepare the license file from an e-mail attachment, complete the following steps:

1. Transfer the license to z/OS, as described in *Transferring a License File from PC to a z/OS Host Using FTP*, in *Mainframe Product Licensing*.
2. Verify that the transferred license file is stored in an Adabas source library (with RECFM=F or FB and LRECL=80), taking care to preserve its format as ASCII.

To prepare the license file from the installation tape, complete the following steps:

- Verify that the license file is stored from the tape into an Adabas source library (with RECFM=F or FB and LRECL=80), taking care to preserve its format as ASCII.

Installing the Product License File

Once the license file has been prepared, you can install it in one of two ways:

- You can convert the license to a load module (ADALIC) that is then loaded by the Adabas nucleus.
- You can reference the license file in the Adabas nucleus startup job by DD statement.

The following describes both methods.

To convert the license file to a load module, complete the following steps:

1 Review and modify sample job ASMLICAM, as follows:

- Change the STEPLIB DD statement to point to the license load library (MLC*vrs*.LOAD).
- Change the SYSUT1 DD statement to point to the data set containing the Adabas license file you transferred to z/OS earlier.
- Specify an appropriate user load library for the L.SYSLMOD DD statement.

Note: This user load library must also be included in the STEPLIB concatenation for the Adabas nucleus (ADANUC).

2 Submit sample job ASMLICAM. This job runs the MAKE function of the LICUTIL utility to convert the license text file to an assembler source module. ASMLICAM then links and assembles the assembler source to generate a load module called ADALIC, which is stored in the specified user load library (L.SYSLMOD DD statement). For more information about the LICUTIL utility, read *Using The License Utility: LICUTIL*, in *Mainframe Product Licensing*

3 Update your Adabas ADANUC nucleus job to reference the user load library so ADALIC will be loaded by the Adabas nucleus, as described in ***Step 5: Customize and Start the Adabas Nucleus and Test Adabas Communications***.

To reference the license file in the Adabas nucleus startup job, complete the following steps:

1 Make sure any previously created ADALIC load module is inaccessible to the Adabas load library being used by your nucleus job. Adabas first tries to load ADALIC and, if unsuccessful, it reads from a DDLIC data set referenced in ADANUC.

2 Update your Adabas ADANUC nucleus jobs to reference the license, as described in ***Step 5: Customize and Start the Adabas Nucleus and Test Adabas Communications***.

Step 2: Allocate and Format the Adabas Database (SMA Job Number I030)

Customize and run the ADAFRM utility job to allocate and format the Adabas database. The following must be customized:

- Data set names for the database and libraries;
- Volumes for libraries and data sets for the database;
- Space allocation for data sets for the database;
- The Adabas SVC number, the database ID, and database device type(s);
- Sizes of the data sets for each ADAFRM statement.

Step 3: Define the Global Database Characteristics (SMA Job Number I030)

Customize and run the ADADEF utility job to define the global definition of the database. The following must be customized:

- Data set names of the database and libraries;
- The Adabas SVC number, the database ID, and database device type(s);
- ADADEF parameters.

Step 4: Load the Demonstration Files (SMA Job Number I050)

Customize and run the job:

- ADALODE to load the sample demo file EMPL;
- ADALODV to load the sample demo file VEHI;
- ADALODM to load the sample demo file MISC; and
- ADALODP to compress and load the sample Personnel (PERC) demo file and its associated LOB demo file.

Note: The Personnel demo file must be installed on a UES-enabled database because it includes wide-character format (W) fields.

For each job, the following items must be customized:

- Data set names for the database and libraries;

- The Adabas SVC number, the database ID, and database device type(s);
- ADALOD parameters.

Step 5: Customize and Start the Adabas Nucleus and Test Adabas Communications (SMA Job Number I040)

Customize and run an appropriate ADANUC job to start the Adabas nucleus. (This processing will also test basic Adabas communications.) The following modifications to the ADANUC job must be made:

1. Our licensing requires that the modules LICMAIN and LICUTIL be loaded when your nucleus starts up. These modules are distributed in the MLC*vrs*.LOAD library. You must either:
   - Copy LICMAIN and LICUTIL into ADA*vrs*.LOAD; or
   - Concatenate MLC*vrs*.LOAD with ADA*vrs*.LOAD. A sample job, ADANUCS, demonstrates how to do this.
2. Verify that the appropriate product/sub product license file is correctly referenced in the ADANUC job.

   Do either of the following:

   - Verify that the license load module(s) installed in ***Step 1: Check, Prepare, and Install the Product License File***, is/are stored in a load library that is accessible to the Adabas load library. Add the user load library in which license load module(s) reside(s) to the STEPLIB concatenation of the Adabas nucleus job or copy the licensing library into ADA*vrs*.LOAD.
   - Verify that there is/are no appropriate product/sub product load module(s) accessible to the Adabas load library and that the following DD statement is included in the ADANUC job for each appropriate product/sub product utilized:

```
//DDLIC     DD  DISP=SHR,DSN=dsn
//DDLAZPAD  DD  DISP=SHR,DSN=dsn
//DDLACF    DD  DISP=SHR,DSN=dsn
//DDLADE    DD  DISP=SHR,DSN=dsn
//DDLAEZ    DD  DISP=SHR,DSN=dsn
//DDLALA    DD  DISP=SHR,DSN=dsn
//DDLALS    DD  DISP=SHR,DSN=dsn
//DDLASM    DD  DISP=SHR,DSN=dsn
//DDLAOS    DD  DISP=SHR,DSN=dsn
```

   where *dsn* is the data set name of the license file loaded from the tape (in ASCII format). Note that *dsn* could reference a member in a partitioned data set.

   Note: Adabas first tries to load the license load module and, if unsuccessful, it reads from the DDL*ccccc* data set.

For a list of license file names, load modules and DD/Link names, refer to *Adabas and Add-on Licenses*, elsewhere in this guide.

3. Data set names for the database and libraries must be customized for your installation.

Note: Be sure to include appropriate user load libraries.

4. The Adabas SVC number, the database ID, and database device type(s) must be customized for your installation.
5. ADARUN parameters must be customized for your installation.

Step 6: Load Adabas Online System Add-On Product, If Appropriate (SMA Job Number I061)

Customize and run the job ADAINPL to load the Adabas Online System (AOS) into a Natural system file using the appropriate batch version of Natural (read the Adabas Online System documentation to determine its Natural requirements). The following items must be customized:

- Data set names of the database and libraries;
- The Adabas SVC number, the database ID, and device type(s);
- The Natural INPL parameters and system file number.

Alternatively, install the AOS demo version delivered with Adabas: see the section *Installing the AOS Demo Version*.

Note: Demo versions of Adabas Vista (AVI), Adabas Fastpath (AFP), Adabas SAF Security (AAF), and Adabas Transaction Manager (ATM) are automatically installed when you install either the demo or full version of AOS. If you are using the full version of any of the above products, use the SYSPROD Natural system command on each full version product to check if you have installed the latest cumulative INPL update from Empower. Install any missing updates as described in the specific product's documentation.

6 Installing Adabas with Workflows

- Before you Install Adabas with z/OSMF Workflows 48
- To Install an Adabas Environment with Workflows 49
- Variables Reference for Workflows 50

Before you Install Adabas with z/OSMF Workflows

z/OSMF Workflows can be downloaded from the Software Download Center (SDC) on our Empower Portal at **https://empower.softwareag.com.**

Before you use z/OSMF Workflows to install a mainframe product, ensure that the following are present:

1. **Access to z/OS UNIX System Services (USS)**

   For the installation, you need access to USS to transfer and store the required workflows and related files.

2. **Access to z/OSMF**

   For the installation, the user needs access to the z/OS Management Facility to run the Workflows application.

3. **Product data sets**

   Product data sets of Adabas, Natural, and Natural TSO Interface must already be present on disk. Otherwise, the user needs to transfer them from the delivery tape to the mainframe system with all required information.

4. **Data sets must be APF authorized**

   The following data sets must be APF authorized:

```
<HLQ-PROD>.ADAvrs.LOAD
<HLQ-PROD>.MLCvrs.LOAD
<HLQ-ENV>.OSMF.LOAD
    ↩
```

5. **License files for Adabas and Natural**

   The following license files are required:

```
ADAvrs.LICS    Adabas license file
AOSvrs.LICS    Adabas Online System license file
NATvrs.LICS    Natural license file
    ↩
```

6. **Existing instance of an Adabas SVC**

This workflow iteration does not install a new SVC. You require an existing SVC, or you need to install a new one before you start with the workflow installation. See the Adabas documentation for information about installing the SVC.

7. **Configured variableInputFile**

   The variable `workflows` utilizes a variable input file to (pre-) configure and save environment values. A sample file *variableInputFile* is part of the delivered workflows package. Although not required, its use is strongly recommended

To Install an Adabas Environment with Workflows

1. Create an Adabas workflow instance.
2. Click **Review Variable Values**.

   If all values of the variable input file are appropriate for your environment, continue with the next steps.
3. Click **Create installation data sets, sources and procedures**.

   The description of the step as well as any expected errors are shown on screen.

   Perform the step to create the required Adabas installation data sets, sources and procedures. Review the output, then close the window to return to the overview of the workflow steps.
4. Click **Format Adabas data sets**.

   In this step, the Adabas utilities are submitted to create the database.

   Follow the guided steps and complete each of them.

   Close the window to return to the overview of the work steps.
5. Perform the steps under **Load Adabas Files** and submit them.
6. (Optional) Perform the steps under **Generate Adabas Report** and submit them.

The Adabas Nucleus is started with a started task and the Adabas installation is completed.

Variables Reference for Workflows

Category	Variable	Description	Possible values
Adabas	ADA-DBID	ID of the physical database used for the installation	1 - 65535
Adabas	ADA-SVCNO	Adabas SVC number used for internal communication	200 - 255
Adabas	ADA-VRS	Version number of Adabas in a three-digit format without separator	861
Adabas	ADA-PREFIX	Prefix for the name of the Adabas started task, stored in the USR-PROCLIB. It has a maximum length of 3.	AAA-ZZZ
Natural	NAT-BP-SIZE	Size of the local Natural buffer pool in KB	256 - 2097151
Natural	NAT-BP-NAME	Name of the Global Buffer Pool. It has a maximum length of 8.	
Natural	NAT-SUBSID	Natural subsystem ID	
Natural	NAT-FNAT	Default Natural System File number for system programs	1 - 65535, except 255
Natural	NAT-FUSER	Natural System File number for user programs	1 - 65535, except 255
Natural	NAT-VRS	Version number of Natural in a three-digit format without separator	923
Natural	NAT-VRS-2D	Version number of Natural in a two-digit format without separator	92
Natural	NAT-ASM-NAME	Name of your Authorized Services Manager. It has a maximum length of 8.	
General	HLQ-ENV	High-level qualifier for the environment which will be used to construct the data set names	
General	HLQ-PRODS	High-level qualifier where the product data sets are located	
General	MLC-VRS	Version number of the Mainframe License Check in a three-digit format without separator	141
General	USR-PROCLIB	Data set name of the proclib where the started task is stored	

7 Post Install and Verification

- Step 1: Test Adabas Address Space Communications, If Appropriate 52
- Step 2: Terminate the Adabas Nucleus 52
- Step 3. Back Up the Database 52
- Step 4: Insert the ADARUN Defaults 53
- Step 5: Install the Required TP Link Routines for Adabas 53

Step 1: Test Adabas Address Space Communications, If Appropriate

Customize and run the job ADAREP in MULTI mode with the `CPEXLIST` parameter to test Adabas address space communications. The following must be customized:

- Data set names for the database and libraries;
- The Adabas SVC number, the database ID, and database device type(s);
- ADAREP parameters.

Step 2: Terminate the Adabas Nucleus

Communicate with the Adabas nucleus to terminate the session either with an `ADAEND` operator command using the OS `Modify` command:

```
F jobname,ADAEND
```

or

```
P jobname
```

where *`jobname`* is the job or task name of the started nucleus.

Step 3. Back Up the Database

Customize and run the ADASAV utility job to back up the database. The following must be customized:

- Data set names of the database and libraries;
- The Adabas SVC number, the database ID, and device type(s);
- ADASAV parameters.

Step 4: Insert the ADARUN Defaults

The member DEFAULTS in the Adabas JCL library can be modified to set the ADARUN defaults. The following must be customized:

- Data set names of the database and libraries;
- ADARUN user defaults:
  - Device type(s) (default: 3390)
  - SVC number (default: 249)
  - Database ID (default: 1)

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

Step 5: Install the Required TP Link Routines for Adabas

Refer to the section ***Installing Adabas With TP Monitors*** for a description of the TP link routine procedure.

8 Installing Adabas Client Link Routines

- Preparing Adabas Link Routines 56
- Installing Adabas with IMS TM 62
- General Considerations for Installing Adabas with CICS 65
- Installing Adabas with CICS 66
- Installing the CICS High-Performance Stub Routine 80
- Installing Adabas with Com-plete 96
- General Considerations for Installing Adabas with Batch/TSO 98
- Installing Adabas with Batch/TSO 99
- Establishing Adabas SVC Routing by Adabas Database ID 105
- Modifying Source Member Defaults (LGBLSET Macro) 113

To establish correct communication between clients and Adabas databases, it is necessary to install Adabas client link routines in each supported client environment.

The table below lists all supported client environments with links to the corresponding installation instructions.

Client Environment	Installation Instructions
CICS	■ General Considerations for Installing Adabas with CICS ■ Installing Adabas with CICS ■ Installing the CICS High-Performance Stub Routine
Batch/TSO	■ General Considerations for Installing Adabas with Batch/TSO ■ Installing Adabas with Batch/TSO
Com-plete	Installing Adabas with Com-plete
IMS TM	Installing Adabas with IMS TM

In addition, the following links provide installation information which is applicable to all the above client environments:

- Preparing Adabas Link Routines
- Establishing Adabas SVC Routing by Adabas Database ID
- Modifying Source Member Defaults (LGBLSET Macro)

Preparing Adabas Link Routines

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

- ADAUSER Considerations
- AMODE/RMODE Assembly Directives in Adabas Link Routines
- Link Routine User Exit 1 (Pre-Command) and User Exit 2 (Post-Command)
- UES-Enabled Link Routines

Important: If an ADALNK batch link routine has been linked or modified by Software GmbH product modules or user exits, it cannot be used in any application startups of Adabas utility jobs or Adabas, Entire System Server, Adabas Review Hub, or Entire Net-Work nuclei.

ADAUSER Considerations

ADAUSER is a module with an entry point ADABAS that is normally link-edited with batch (single tasking and multi-tasking) application programs.

On the first Adabas call, ADAUSER dynamically loads and calls the ADARUN module which establishes the appropriate Adabas link module to be used for calling Adabas. Subsequent Adabas calls bypass ADARUN and are routed directly to the Adabas link module.

When ADARUN is called it processes its DDCARD control statements. For the ADARUN control statement `PROGRAM=USER` (the default), ADARUN loads the non-reentrant Adabas link module ADALNK. For applications that require use of the reentrant link module ADALNKR, the ADARUN control statement `PROGRAM=RENTUSER` should be used.

Note: When using ADARUN `PROGRAM=RENTUSER`, for the reentrant link module ADALNKR to honor the use of the ADARUN control statements DBID= and SVC=, the ADARUN control statement LNKGNAME= should be used to provide a non-reentrant LNKGBLS module for ADARUN to propagate the DBID= and SVC= values.

AMODE/RMODE Assembly Directives in Adabas Link Routines

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

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

This section covers the following topics:

- Re-linking Adabas Link Routines
- ADAUSER AMODE/RMODE Considerations

Re-linking Adabas Link Routines

When re-linking the Adabas 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 link routines all run AMODE(31) after initialization, but they will return to the caller in the caller's AMODE.

Note: Under CICS, the Adabas link routines run AMODE(31), but the Dataloc RDO parameter governs the AMODE and RMODE of the running CICS transaction.

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

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

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

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

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

ADAUSER AMODE/RMODE Considerations

ADAUSER has AMODE(31), RMODE(ANY). This is the most flexible configuration for linking ADAUSER with the widest variety of application programs. However, if an AMODE 24 application program loads ADAUSER dynamically, the ADAUSER module should be re-linked to AMODE(31), RMODE(24) to ensure that the program can invoke ADAUSER properly below the 16-megabyte line.

Link Routine User Exit 1 (Pre-Command) and User Exit 2 (Post-Command)

A pre-command user exit and a post-command user exit may be linked with the Adabas link globals table for subsequent invocation by the Adabas link module. Refer to the LGBLSET parameters USERX1 and LX1NAME, and USERX2 and LX2NAME respectively for information on how to ensure any linked exits are invoked.

User exit 1 receives control *before* a command is passed to a target.

Note: Special commands emanating from utilities and from Adabas Online System are marked as physical calls. These calls must be bypassed in user exits. These calls have X'04' in the first byte (TYPE field) of the command's Adabas control block (ACBX). Any link routine user exit 1 must check this byte and return if it is set to X'04'. Be sure to reset R15 to zero on return.

User exit 2 receives control *after* a command has been processed by a target, the Adabas SVC, or the link module itself.

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

Register	Contents
1	Address of the User Block (UB). Refer to User Buffers for information regarding the User Block fields which describe the length and location of the user information area when configured using the LGBLSET LUINFO parameter.
2	Address of an 18-word format 1 register save area.
13	For CICS, on entry to the link user exit, R13 points to the CICS DFHEISTG work area.
14	Return address.
15	Entry point address: User exit 1 or 2.

Any registers except register 15 that are modified by the user exits must be saved and restored; the address of a save area for this purpose is in register 2 on entry to the exit.

If at return from user exit 1, register 15 contains a value other than zero (0), the command is not sent to the target but is returned to the caller. The user exit should have set a non-zero response code in the Adabas control block to indicate to the calling program that it has suppressed the command: response code 216 (ADARSP216) is reserved for this purpose.

UES-Enabled Link Routines

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

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

This section covers the following topics:

- Default or Customized Translation Tables
- Calling LNKUES
- Sample Jobs for Universal Encoding Support
- Disabling UES Support

Default or Customized Translation Tables

By default, the load modules for all Adabas 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 LNKUES module is functionally reentrant; however, it is not linked that way in the Adabas load library.

3. When linking the LNKUES load module and the translation tables, the linkage editor may produce warning messages concerning the reentrant or reusability status of the linked module. These warning messages can be ignored.
4. If relinking an Adabas link routine for UES support, the LNKUES module must be included. This will ensure that your new Adabas applications have support for Adabas 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 requests, LNKUES receives control before UEXIT1. In Adabas replies, LNKUES receives control after UEXIT2.

Sample Jobs for Universal Encoding Support

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

Sample Job	Location	Description
LNKGCICS	ACIvrs.JOBS	Links the CICS globals table with LNKUES and the default translation tables ASC2EBC and EBC2ASC.
LNKLCO8	ADAvrs.JOBS	Links the Com-plete link globals table with LNKUES and the default translation tables ASC2EBC and EBC2ASC.
LNKLNI8	AIIvrs.SRCE	Links the IMS link routine with the LNIGBL link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.
LNKLNK8	ADAvrs.JOBS	Links the batch link routine with the LNKGBLS link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.
LNKLNKR8	ADAvrs.JOBS	Links the reentrant batch link routine with the LNKRGBL link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.

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

Note: If you use link routine user exits, Adabas Review, or Adabas System Coordinator, the jobs used to link these components with the batch, reentrant batch or the IMS link routine should be modified to explicitly include the LNKIND module when the link routines are relinked to incorporate user or Software GmbH product link routine exits.

Disabling UES Support

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

To disable UES support in link routines:

1. Edit the link globals table for the associated link routine. Set the UES parameter setting to NO.
2. Assemble the link globals table after making any other necessary modifications to the equates and other directives in the source module as required by your installation.
3. Link the Adabas link routine with the newly assembled link globals table and do not include any of the UES components (that is, LNKUES, ASC2EBC, or EBC2ASC).

Installing Adabas with IMS TM

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

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

Hint:

For a COBOL IMS message processing program (MPP) to call 'ADABAS' USING …" in an IMS environment you must link ADALNI to the application that runs in the message processing program (MPP). Since ADALNI is non-reentrant, a link of ADALNI will make the application non-reentrant too. To resolve this:

- Add ALIAS ADABAS to the link of ADALNI.
- Add the load module name ADABAS to the preload list of the MPP.
- Add the load library that contains ADABAS to the steplib chain of the MPP.
- Use the DYNAM option of COBOL to force a dynamic load of ADABAS for the CALL 'ADABAS' statement.

Note: If CALL 'ADABAS' results in a dynamic load of the Adabas link routine, the load module name of the Adabas link routine must be called ADABAS. As ADALNI saves the

address of its working storage in itself, ADALNI must never be unloaded in the MPP. Otherwise you will get memory leaks.

This section covers the following topics:

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

IMS TM Link Routines

These are Adabas link routines for IMS TM:

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

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

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

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

Obtaining the Adabas User ID

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

Obtaining the SAF ID

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

Installation Procedure for IMS

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

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

   Note: The OPSYS parameter must be set to ZOS.

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

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

   A new link globals module (with the name specified by the GBLNAME parameter in LNIGBL) will be generated in the user load library identified in the ASMGBLS job.
4. Copy sample job LNKLNI8 to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the ADALNI8 base module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from this job to an appropriate user load library. LNKLNI8 can be found in the AII*vrs*.SRCE library.

   Note: If you use link routine user exits, Adabas Review, or Adabas System Coordinator, the jobs used to link these components with the batch, reentrant batch or the IMS link routine should be modified to explicitly include the LNKIND module when the link routines are relinked to incorporate user or Software GmbH product link routine exits.

   The module resulting from this job is ADALNI.
5. Place the ADALNI module in a load library available for IMS MPP regions.

   The Adabas link routine is prepared.

General Considerations for Installing Adabas with CICS

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

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

CICS MRO Environment Requirements

If you run with the CICS multiple region option (MRO), you must set the `MRO` parameter to "YES" and use the default for the `NETOPT` parameter. These parameters are supplied via the LGBLSET macro (read ***Modifying Source Member Defaults (LGBLSET Macro)***, elsewhere in this section).

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

Sample Resource Definitions

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 components. Consult the appropriate IBM CICS documentation for information on the DFHCSDUP utility. The DEFADAC member can be found in the ACI*vrs*.SRCE library.

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

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

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

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

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

This section covers the following topics:

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

The Adabas CICS Application Stub (ADACICS)

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

The Adabas CICS Names Module (ACINAMES)

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

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

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

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

The Adabas CICS Installation Options Table (ACIOPT)

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

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

The MACINS Macro

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

The MACINS macro and a sample ACINAMES source member are provided in the ACI*vrs*.SRCE library.

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 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 and sample ACIOPT source member are provided in the ACI*vrs*.SRCE library.

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.	ACIOPT

Parameter	Description	Default
	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.	
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
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. 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	8

Parameter	Description	Default
	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.	

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
```

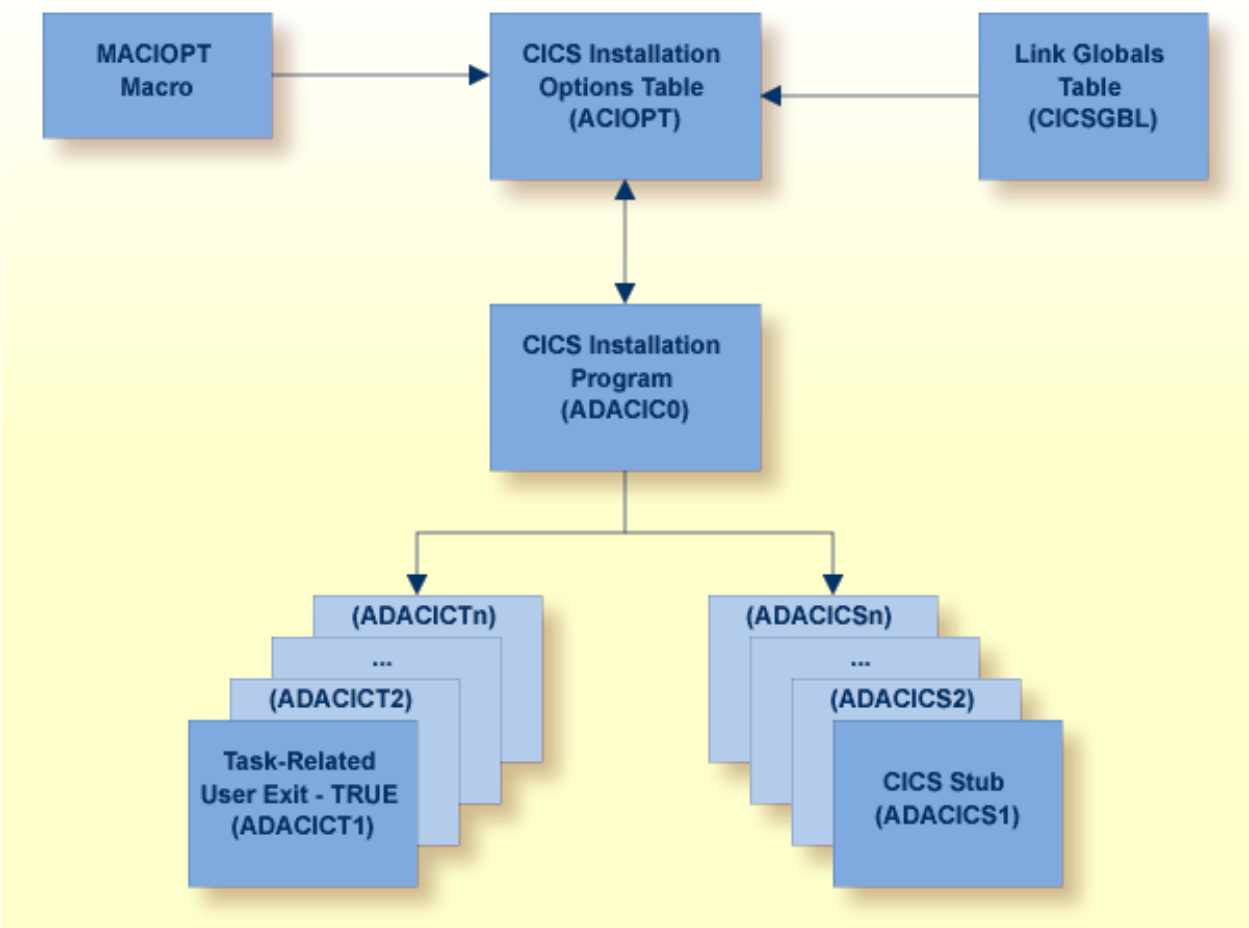
Adabas Task-Related User Exits (TRUEs)

In a simple Adabas CICS transaction that uses the EXEC CICS LINK command to communicate with Adabas, there should be one invocation of the Adabas Task Related User Exit (TRUE) for each EXEC CICS LINK issued from the application.

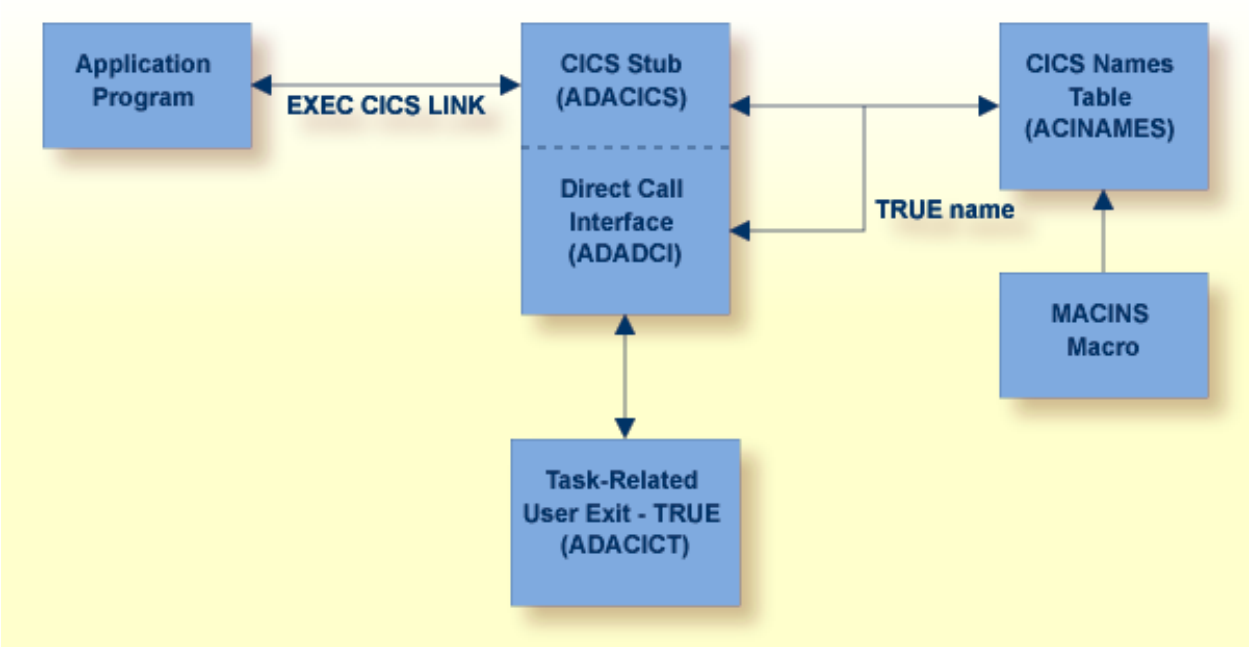
If the Adabas CICS interface employs exits such as the Adabas Fastpath exit or other System Co-ordinator facilities, there may be more than one invocation of the Adabas TRUE for each EXEC CICS LINK issued by the application program. Other Software GmbH products that can have multiple TRUE invocations for each LINK to Adabas are the Adabas Bridge for DL/I and Natural. If the Adabas high-performance stub (BALR interface) is employed by applications, including Natural, there will be multiple invocations of the Adabas TRUE for each EXEC CICS LINK to the Adabas interface module.

Adabas supports 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 Objects

The following table lists the objects supplied to support the installation of Adabas with CICS.

Module	Location	Description
ADACIC0	ACI*vrs*.LOAD	Adabas CICS installation program
ADACICS	ACI*vrs*.LOAD	Adabas CICS Application Stub.
ADACICT	ACI*vrs*.LOAD	Adabas CICS task-related user exit (TRUE).
ASMCINS	ACI*vrs*.JOBS	Sample job to assemble the ACINAMES module and link it with the Adabas CICS application stub.
ASMCOPT	ACI*vrs*.JOBS	Sample job to assemble and link the ACIOPT source module as the standalone module named ""ACIOPT"". See **Step 2** of the ***Installation Procedure***.
ASMGBLS	ADA*vrs*.JOBS	Sample job to assemble and link the Adabas globals table. See **Step 7** of the ***Installation Procedure***.
CPYCICSM	ACI*vrs*.JOBS	Sample job to copy the distributed load modules to a DFHRPL concatenated library. See **Step 1** of the ***Installation Procedure***.
DEFADAC	ACI*vrs*.SRCE	Sample CICS DEFINE control statements. See **Step 9** of the ***Installation Procedure***.
LNKATRU	ACI*vrs*.JOBS	Sample job to link the Adabas CICS task-related user exit. See **Step 3** of the ***Installation Procedure***.
LNKGCICS	ACI*vrs*.JOBS	Sample job to relink the Adabas globals table. See **Step 8** of the ***Installation Procedure***.

Installation Procedure for CICS

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

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

Step 1. Copy the Load Modules

Copy the Adabas CICS load modules from the Adabas distribution library to a load library that will be in the CICS DFHRPL concatenation (see sample member CPYCICSM in the ACI*vrs*.JOBS library).

Step 2. Prepare the Adabas CICS Installation Options Table

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

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

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

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

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

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

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

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

The MACIOPT macro is located in the ACI*vrs*.SRCE library. For complete information on the MACIOPT macro, read ***The MACIOPT Macro***, elsewhere in this section. A sample source member ACIOPT is provided in the ACI*vrs*.SRCE library.

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

The ACIOPT module may be defined to CICS using the CEDA/RDO facility or the DFHCSDUP utility. Sample DFHCSDUP statements are provided in the DEFADAC member in the ACI*vrs*.SRCE library.

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

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

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

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

1 Relink the ADACICT module with a NAME or PHASE statement giving a new name for each Adabas TRUE.

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

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

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

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

To prepare the ACINAMES module, complete the following step:

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

  The MACINS macro is provided in the ACI*vrs*.SRCE library.

Sample job, ASMCINS, which is provided in the ACI*vrs*.JOBS library, assembles the ACINAMES module and links it with the Adabas CICS application stub (see **Step 5**), and names the stub "ADABAS".

Effective with Adabas 8.6 and later versions, the CICS macro library is required to assemble ACINAMES.

Example

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

```
*        Sample “ACINAMES” for Adabas 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.

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

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

  Sample job, ASMCINS, which is provided in the ACI*vrs*.JOBS library, assembles the ACINAMES module and links it with the Adabas CICS application stub (see **Step 4**), and names the stub "ADABAS".

Example

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

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

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

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

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

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

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)***, elsewhere in this section.

- 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.
- In CICS environments only, if you want your security system user IDs to be stored in Adabas user queue elements (making them available for display and review as well as preventing response code 200, ADARSP200, subcode 21 when ADARUN SECUID=REQUIRE is in effect for Adabas), you must code the SAF parameter as YES. This is only required in CICS environments; in other environments, the security system user IDs are automatically stored.

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

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

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

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

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

Step 9. Modify CICS Installation Values -- DEFADAC

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

Step 10. Update the CICS CSD File

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

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

Modify the CICS PLTPI table to add an entry for the Adabas CICS installation program ADACIC0. The Adabas CICS installation program will be called by CICS to start the Adabas CICS execution units while CICS is starting up. Use member ADAPLTXX from the ACI*vrs*.SRCE library as a sample.

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

Step 12. Start the CICS

Start the CICS and note any ADAK* messages relating to the execution of the Adabas CICS installation program that appear on the console.

Installing the CICS High-Performance Stub Routine

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

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

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

The high-performance stub routine is written in Assembler language. When linked with the application program, it serves as an interface between the application and the Adabas CICS link

component. The application program can then issue CALL statements to access the stub routine when executing an Adabas command.

A CICS/TS application derives the following advantages from the high-performance stub:

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

This section covers the following topics:

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

Restrictions and Requirements

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

1. The Adabas high-performance stub routine is available for all supported versions of CICS/TS.

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

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

2. Supported Programming Languages

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

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

Stub Components

Location	Member	Description
ACI*vrs*.SRCE	ADAGSET ALCSIVP COBSIVP LNCSTUB	macro required for assembling LNCSTUB and ALCSIVP source for the ALC install verification source for the COBOL install verification source for the high-performance stub
ACI*vrs*.JOBS	JCLALCI JCLCOBI JCLLNCS	sample JCL for ALC install verification sample JCL for COBOL install verification sample JCL for LNCSTUB (high-performance stub)

Installation Overview

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

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

This procedure is described in the following steps:

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

Step 1: Install the LNCSTUB Module

The Adabas CICS high-performance stub routine is an Assembler language source module, provided in member LNCSTUB in the ACI*vrs*.SRCE library.

Step 1 has the following substeps:

Edit the ADAGSET Macro

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

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

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

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

(Optional) Set the LNCSTUB Entry-Point Alias

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

Modify Member JCLLNCS

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

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

Value	Description
&NCAL	Value for the linkage editor `NCAL` parameter. The recommended value is NCAL.
&LSIZE	Primary and secondary table sizes used by the linkage editor.
&WORK	DASD device type to use for temporary and utility data sets.

Preprocess, Assemble, and Link the LNCSTUB Module

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

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

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

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

```
NAME LNCSTUB(R)
```

- Do not include the CICS stub modules DFHEAI0 & DFHEAI1 with the LNCSTUB load module. As a result, however, the following occurs:
  - The linkage editor issues IEW462 or similar messages indicating that DFHEAI1 is an unresolved external reference;
  - The LNCSTUB module may be marked NOT EXECUTABLE by the linkage editor;
  - A condition code of 8 may be set in the link step.

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

Make the LNCSTUB Available to Application Programs

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

The LNCSTUB module has either an entry name of LNCSTUB or the alias entry name as coded in the SETC statement to set the value of &STBNAME. Either value may be used by the application program as the object of a CALL statement to pass control to LNCSTUB with a list of parameters. The language-specific calling conventions for LNCSTUB are discussed later in this section.

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

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

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

Step 2: (Optional) Install and Execute an IVP

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

This section describes each of these IVPs:

Note: The two installation verification programs ALCSIVP and COBSIVP only use fields AA and AE from the provided demonstration EMPLOYEES file. For more information about the provided demonstration files, read ***Load the Demonstration Files*** in the installation instructions, provided elsewhere in this guide.

Install and Execute the Assembler IVP: ALCSIVP

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

To install and execute the Assembler IVP, ALCSIVP:

1 Modify the source member ALCSIVP in ACI*vrs*.SRCE:

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

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

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

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

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

2 Modify the sample job stream, JCLALCI in ACI*vrs*.JOBS:

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

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

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

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

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

                    DEFINE TRANSACTION(STB1) GROUP(ADABAS)
                    DESCRIPTION(TRANSACTION TO EXECUTE THE ASSEMBLER IVP FOR ↩
HIGH-PERFORMANCE STUB)
                  PROGRAM(ALCSIVP) TWASIZE(32) PROFILE(DFHCICST) STATUS(ENABLED)
                    TASKDATALOC(ANY) TASKDATAKEY(USER) STORAGECLEAR(NO)
                   RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
                  PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO) INDOUBT(BACKOUT)
                    RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
                    RESSEC(NO) CMDSEC(NO)
```

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

Install and Execute the COBOL IVP: COBSIVP

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

To install and execute the COBOL IVP, COBSIVP:

1 Modify the source member, COBSIVP in ACI*vrs*.SRCE:

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

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

2 Modify the sample job stream, JCLCOBI in ACI*vrs*.JOBS:

Member JCLCOBI is used to preprocess, compile, and link the COBSIVP installation verification program. To modify the JCLCOBI example to meet site requirements, change the JOB card in the member and provide values for the symbolic procedure variables as described in the following table:

Value	Description
&ADALIB	Adabas load library used to provide the ADASTWA load module for the linkage editor.
&MEM	Member name to be processed; in this case, COBSIVP.
&CICSLIB	CICS RPL library where the COBSIVP load module is placed for execution under CICS.
&COBLIB	COBOL compiler STEPLIB.
&INDEX	High-level qualifier for the CICS macro library used in the SYSLIB DD statement for the compiler.

Value	Description
&INDEX2	High-level qualifier for the CICS load library to use for the translator STEPLIB DD statement, and for the SYSLIB in the link step.
&LINKLIB	COBOL LINKLIB.
&STBSRCE	Source library containing the distributed Adabas CICS high-performance stub LNCSTUB.
&STUBLIB	A load library to contain the LNCSTUB load module. This library should be available to your application programs when they are linked.
&SYSMSG	Output class for translator messages.
&SYSOUT	Output class for SYSOUT and SYSPRINT messages.
&WORK	DASD device type to use for temporary and utility data sets.

3 Preprocess, compile, and link COBSIVP:

Use the modified JCLCOBI job to preprocess, compile, and link the COBSIVP program.

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

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

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

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

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

                    DEFINE TRANSACTION(STB2) GROUP(ADABAS)
                    DESCRIPTION(TRANSACTION TO EXECUTE THE COBOL IVP FOR ↩
HIGH-PERFORMANCE STUB)
                  PROGRAM(COBSIVP) TWASIZE(32) PROFILE(DFHCICST) STATUS(ENABLED)
                    TASKDATALOC(ANY) TASKDATAKEY(USER) STORAGECLEAR(NO)
                   RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
                  PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO) INDOUBT(BACKOUT)
                    RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
                    RESSEC(NO) CMDSEC(NO)
```

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

Step 3: Link and Execute the Application Program

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

Step 3 has the following substeps:

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

Guidelines for Modifying the Application Program

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

- Programming Languages Supported by LNCSTUB

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

- Use of the CICS Transaction Work Area

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

- Use of the CICS COMMAREA

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

- Reentrant Requirement

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

- CICS Requests Issued by LNCSTUB

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

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

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

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

- DCI Entry Point Address

  An EXEC CICS LINK command is issued by LNCSTUB at least once to acquire the DCI entry point from the Adabas CICS link component program. This address is then used for BALR access on all subsequent Adabas calls for a transaction. Thus, the calling application program must provide a fullword (4-byte) field to hold the DCI entry point address obtained by LNCSTUB. This 4-byte field is the first parameter passed to the LNCSTUB module by the call mechanism. The remaining parameters comprise the Adabas parameter list needed to execute an Adabas request. (Either a version 7 or version 8 parameter list may be used)

- DCI Parameter List

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

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

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

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

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

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

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

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

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

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

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

- Restrictions on Application Program Coding

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

- Standard Batch Call Mechanism Used

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

Link the Application Programs to Include the LNCSTUB Module

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

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

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

For linking Assembler language programs:

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

```
INCLUDE DFHEAI
```

The Assembler object input is similar to:

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

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

For linking COBOL language programs:

- For a COBOL program, the SYSLIN input is similar to:

```
REPLACE DFHEAIO
INCLUDE SYSLIB(DFHECI)
```

The COBOL object input is similar to:

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

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

For linking PL/I and C language programs:

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

Performance Using LNCSTUB

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

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

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

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

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

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

Modifying Source Member Defaults (ADAGSET Macro)

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

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

The applicable ADAGSET parameter options, with their default values (underlined), are described below (all other ADAGSET parameters are obsolete and will be removed in a future version):

- ENTPT: Name of the Adabas CICS Application Stub
- LOGID: Default Logical Database ID
- PARMTYP: Area for Adabas Parameter List

ENTPT: Name of the Adabas CICS Application Stub

Parameter	Description	Syntax
ENTPT	The name of the Adabas CICS Application Stub used in the priming EXEC CICS LINK command issued by LNCSTUB.	ENTPT={'ADACICS' | 'name'}

LOGID: Default Logical Database ID

Parameter	Description	Syntax
LOGID	The database ID used by the (optional) Assembler IVP, ALCSIVP.. Valid ID numbers are 1-65535.	LOGID= *nnn*

PARMTYP: Area for Adabas Parameter List

Parameter	Description	Syntax
PARMTYP	The area which is to contain the Adabas parameter list. If PARMTYP=TWA is specified the CICS TWA is used, otherwise the CICS COMMAREA is used. This value should match that specified for the LGBLSET parameter of the same name in use by the Adabas CICS Application Stub.	PARMTYP={ ALL | COM ↩ | TWA }

Installing Adabas with Com-plete

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

Software GmbH's TP monitor, Com-plete requires an Adabas link routine if it is to communicate with Adabas databases, use Software GmbH's Entire Net-Work product, or use products like Entire System Server running under Com-plete.

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

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

Module	Description
ADALCO8	Base module
ADALCO	Executable default module

To prepare the Adabas link routine:

1 Copy sample member LCOGBL provided in the ADA*vrs*.SRCE library to any appropriate user source library where it can be modified (where *vrs* is the number of the latest Adabas *version* delivered on the tape). LCOGBL is a module containing LGBLSET parameters that are used to create default settings for Adabas link components. A complete description of LGBLSET parameters can be found in ***Modifying Source Member Defaults (LGBLSET Macro)***, elsewhere in this guide.

2 Modify the LCOGBL member in the user source library.

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

Parameter	Specify...
LOGID	The default database or target ID. This should be a numeric value between "1" and "65535". The default value is "1". **Note:** Specifying default values for LOGID and SVCNO under Com-plete is for documentation purposes only. The ADASVC Com-plete runtime control statements will provide the supported database ID/Adabas SVC combinations to be used in running Com-plete. For more information, read the Com-plete documentation.

Parameter	Specify...
SVCNO	The default Adabas SVC number. This number should be between "200" and "255". **Note:** Specifying default values for LOGID and SVCNO under Com-plete is for documentation purposes only. The ADASVC Com-plete runtime control statements will provide the supported database ID/Adabas SVC combinations to be used in running Com-plete. For more information, read the Com-plete documentation.
TPMON	COM. This keyword specifies the three-character TP monitor abbreviation. For Com-plete, this value should be "COM".
RENT	YES. This keyword indicates whether or not the module is serially reentrant. For Com-plete, this value should be "YES".
GEN	CSECT. This keyword indicates whether a CSECT or DSECT is generated. CSECT must be specified so an object module is generated that can be linked as the link routine globals load module.
UES	Whether Adabas Universal Encoding Support (UES) should be enabled. The default is YES. For more information, read ***Enabling Universal Encoding Support (UES) for Your Adabas Nucleus***, elsewhere in this guide.
exit parameters	Whether any other exits are to be active, and in the case of user exits you provide, specify the user exit module names. Specify this information in other parameters of LGBLCOM, as described in ***Modifying Source Member Defaults (LGBLSET Macro)***, elsewhere in this guide.

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

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

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

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

   The module resulting from this job is called ADALCO.

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

   The Adabas link routine is prepared.

General Considerations for Installing Adabas with Batch/TSO

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

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

However, it is important to note that user programs linked with ADAUSER also load ADARUN. ADARUN, in turn, loads other modules.

To start a user program linked with ADAUSER, the following modules must all be available from the defined load libraries for that specific TSO user at execution time:

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

This section covers the following topics:

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

Important: If an ADALNK batch link routine has been linked or modified by Software GmbH product modules or user exits, it cannot be used in any application startups of Adabas utility jobs or Adabas, Entire System Server, Adabas Review Hub, or Entire Net-Work nuclei.

Non-reentrant ADALNK Batch Routine Operation

The ADALNK module in the Adabas load library operates when the following conditions are met:

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

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

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

ADALNKR: Reentrant Batch Link Routine

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

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

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

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

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

Installing Adabas with Batch/TSO

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

This section covers the following topics:

- Supplied Modules
- Changing Default Values for the ADALNK or ADALNKR Modules
- Zapping the Default Values for the ADALNK or ADALNKR Modules
- Enabling TCP/IP Socket Support for Batch/TSO

Important: If an ADALNK batch link routine has been linked or modified by Software GmbH product modules or user exits, it cannot be used in any application startups of Adabas utility jobs or Adabas, Entire System Server, Adabas Review Hub, or Entire Net-Work nuclei.

Supplied Modules

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

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

Changing Default Values for the ADALNK or ADALNKR Modules

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

To change default values, complete the following steps:

1. Copy the sample member LNKGBLS (for non-reentrant links) or LNKRGBL (for reentrant links) members provided in the ADA*vrs*.SRCE library to any appropriate user source library where they can be modified. These modules contain LGBLSET parameters that are used to create default settings for link components. A complete description of LGBLSET parameters can be found in ***Modifying Source Member Defaults (LGBLSET Macro)***, elsewhere in this guide.
2. Modify the LNKGBLS or LNKRGBL member in the user source library. Provide values for the LOGID, SVC, and other keywords to suit your installation requirements.
3. Modify and run sample job ASMGBLS as described at the top of the job. ASMGBLS can be found in the ADA*vrs*.JOBS library. When fully modified, the SET statement in the job should reference the LNKGBLS or LNKRGBL member you prepared in the previous step and the NAME link edit control statement should reference the name specified by the GBLNAME parameter in the LNKGBLS or LNKRGBL member.

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

   A new link globals module (with the name specified by the GBLNAME parameter in the LNKGBLS or LNKRGBL member) will be generated in the user load library identified in the ASMGBLS job.
4. Copy sample job LNKLNK8 or LNKLNKR8 (reentrant) to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the appropriate ADALNK8 or ADALNKR8 (reentrant) base module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from the job to an appropriate user load library. LNKLNK8 and LNKLNKR8 can be found in the ADA*vrs*.SRCE library.

Note: If you use link routine user exits, Adabas Review, or Adabas System Coordinator, the jobs used to link these components with the batch, reentrant batch or the IMS link routine should be modified to explicitly include the LNKIND module when the link routines are relinked to incorporate user or Software GmbH product link routine exits.

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

5 Tailor the ADARUN DDCARD input for the job steps that will use the batch/TSO link routines. The DDCARD input should include the following updates:

- Specify the ADARUN PROG=USER parameter for a non-reentrant link routine, or specify ADARUN PROG=RENTUSER to use a reentrant link routine in the job step. For more information about the PROG parameter, read *PROGRAM : Program to Run* in *Adabas Operations Manual*.

6 Make sure the appropriate load libraries are made available to the job step. These may be STEPLIB, TASKLIB, JOBLIB, or, for reentrant modules, the LPA or LINKLIB.

Zapping the Default Values for the ADALNK or ADALNKR Modules

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

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

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

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

Enabling TCP/IP Socket Support for Batch/TSO

This section covers the following topics:

- Prerequisites for TCP/IP socket support
- Supplied Components for TCP/IP socket support
- Enabling Procedure for TCP/IP socket support
- Limitations and Restrictions of TCP/IP socket support

Prerequisites for TCP/IP socket support

Prerequisite to the enabling of TCP/IP socket support for Batch/TSO is the following:

- **Direct TCP/IP access to Adabas nuclei**
- **Adabas Directory Server**

Direct TCP/IP access to Adabas nuclei

Refer to **Enabling Direct TCP/IP Access (ADATCP) To Your Adabas Nucleus** for information on how to enable your Adabas nuclei to accept direct TCP/IP access.

Adabas Directory Server

The Adabas Directory Server provides all necessary information required to accomplish communication between Batch/TSO jobs and Adabas nuclei.

Refer to the Adabas Directory Server documentation for information on how to install Adabas Directory Server on z/OS.

When installed, the location of the Adabas Directory Server is derived from the values defined to the LTCPSET parameters ADIHOST and ADIPORT.

Supplied Components for TCP/IP socket support

The following table lists the components supplied to support the enabling of TCP/IP socket support for Batch/TSO:

Dataset	Member	Description
MVSJOBS	ASMANCR	Sample job to assemble and link the LTCPSET TCP/IP parameter module (TCPANCR).
	TCPLNK TCPLNKR	Sample jobs to link ADALNK/ADALNKR with the TCP/IP communications interface module LNKTCP.
MVSLOAD	LNKTCP	TCP/IP communications interface module.
MVSSRCE	LTCPSET	Macro for creating the TCPANCR module from the defined TCP/IP parameters.
	TCPGBLS	Sample TCP/IP parameters.

Enabling Procedure for TCP/IP socket support

To enable the use of TCP/IP socket support for Batch/TSO, complete the following steps:

- **Step 1. Create a new Batch link globals table with LGBLSET TCPIP=YES**
- **Step 2. Create the TCP/IP parameter module TCPANCR**
- **Step 3. Include LNKTCP in link-edit for ADALNK/ADALNKR**
- **Step 4. Make all applicable modules available to the Batch/TSO job**

Step 1. Create a new Batch link globals table with LGBLSET TCPIP=YES

To enable the use of TCP/IP socket support, the LGBLSET parameter **TCPIP: TCP/IP Support** must be set to Yes.

Step 2. Create the TCP/IP parameter module TCPANCR

First, modify the sample TCPGBLS member in the Adabas source library, and supply values for the following parameters:

Parameter	Description	Syntax
ADIHOST	This parameter specifies the hostname of the Adabas Directory Server (ADI). The hostname is used to attempt to acquire the TCP/IP address of the system where the Adabas Directory Server resides. If used, ADIHOST and ADIPORT must both be specified. Valid values are 1-255 characters with no embedded blanks.	ADIHOST={hostname}
ADIPART	This optional parameter specifies the partition name to be used with the Adabas Directory Server (ADI). The partition name is used to restrict database access by TCP/IP; when the Adabas link module queries ADI for a database target and specifies a partition name, only database entries with the same partition name are eligible for access by TCP/IP. Likewise, if the query does not specify a partition name, only database entries that do not have a partition name are eligible Valid values are 1-32 characters with no embedded blanks.	ADIPART={partition ↩ name}
ADIPORT	This parameter specifies the port number used to communicate with the Adabas Directory Server (ADI). If used, ADIHOST and ADIPORT must both be specified. Valid values are 1-65535.	ADIPORT={port number}
RCVTIME	This is the number of seconds that a task will wait for a reply. If RCVTIME=0 the wait is indefinite. If RCVTIME > 0 and the wait exceeds the specified value, a RSP218 sub-code 130 is returned to the caller.	RCVTIME={0-134217727}

Parameter	Description	Syntax
MSGLVL	This parameter controls what LNKTCP* messages are written to the console. The default value of 0 means no messages are written. Refer to *LNKTCP* Adabas Link Routine TCP/IP Socket Support Messages* to determine what messages may be written for a particular MSGLVL value.	0-255
TRACE	Indicates whether or not tracing should be written out to TCPPRNT. (DD statement TCPPRNT is required in the Batch/TSO job) **Note:** This should only be used when directed to do so by Software GmbH.	TRACE={NO|YES}

Note: The location of the Adabas Directory Server is derived using the following order:

1. Use the supplied ADIHOST and ADIPORT values.
2. Resolve SAGXTSDSMF and SAGXTSDSMFPORT using DNS services.
3. Use localhost with a default port of 4952.

Next, modify and run sample member ASMANCR in the Adabas jobs library as described at the top of the job. This job will assemble and link-edit the TCP/IP parameter module TCPANCR using the TCPGBLS source member as input.

Step 3. Include LNKTCP in link-edit for ADALNK/ADALNKR

With reference to the sample members TCPLNK and TCPLNKR in the Adabas jobs library, modify your existing ADALNK/ADALNKR jobs to include the module LNKTCP.

Step 4. Make all applicable modules available to the Batch/TSO job

Ensure the following modules are available to the Batch/TSO job:

- Link globals table with LGBLSET parameter TCPIP=YES.
- TCP/IP parameter module TCPANCR.
- ADALNK/ADALNKR with LNKTCP included.

Limitations and Restrictions of TCP/IP socket support

Currently, the following limitations and restrictions exist in the TCP/IP socket support for Batch/TSO:

1. Adabas commands issued to an Adabas database protected by Adabas SAF Security using TCP/IP socket support are allowed or disallowed based on either the User ID of the user who started the Adabas nucleus or on the identifier of the nucleus started task, and not on the User ID of the user issuing the Adabas command.
2. TCP/IP socket support should not be used when performing SYS*xxx* administration on an Adabas nucleus, Adabas Audit Server, or Adabas Event Replicator Server - doing so may result in response code 218 subcode 113.

Establishing Adabas SVC Routing by Adabas Database ID

Your application programs that use Adabas link routines 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 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 applic-

ation 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***, elsewhere in this section.
2. Assemble and link-edit the DBID/SVC routing table member to create a load module 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)***, elsewhere in this guide. For information on assembling and link-editing the link globals table once the table is updated, refer to the instructions for each 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 modules are included in the job.

This section covers the following topics:

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

- Installing DBID/SVC Routing under CICS

Installing DBID/SVC Routing under Batch, TSO and IMS

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

To install the Adabas DBID/SVC routing feature under 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***, elsewhere in this section.
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*, in *Adabas Installation for z/OS*; for IMS, read *Installing Adabas with IMS TM*, in *Adabas Installation for z/OS*.
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		
	Batch	TSO	IMS
ADALNK	LNKLNK8	LNKLNK8	---
ADALNKR	LNKLNKR8	LNKLNKR8	---
ADALNI8	---	---	LNKLNI8

Installing DBID/SVC Routing under CICS

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

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***, elsewhere in this section.
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 sub-program 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 DEFADAC 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*.

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 CICS Installation Program 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 CICS task-related user exit

(TRUE) module, ADACICT, and is made available on each application call to the TRUE by the Adabas application stub ADACICS.

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, 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 also provides the default database ID and Adabas SVC number used for an Adabas 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

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}]
```

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

MDBSVC TYPE=GEN Syntax

The syntax for the MDBSVC TYPE=GEN statement is:

```
MDBSVC TYPE=GEN,SVC=svcno,DBID={id | (id1,id2[,idn]...) }[,DBID2={id | ↩
(id1,id2[,idn]...) } ]
```

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

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 is required 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. 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. 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. The OPSYS parameter must be set to "ZOS".

PREFIX

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

SVC

The SVC parameter is required 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. 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)

The 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 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
- AUDIT: Adabas Auditing Support
- AVB: Adabas Bridge for VSAM Support
- CITSNM: Adabas CICS TS Queue Name
- COR: Adabas System Coordinator Support
- DBSVCTN: DBID/SVC Routing Table
- DYNDBSVC: DBID/SVC Routing Table
- ENTPT: Name of the Adabas CICS Command-Level Link Routine
- FIXLOGID: Prevent Runtime Modification of LOGID
- GBLNAME: Name of Link Globals Module
- GEN: Generate CSECT or DSECT
- LMINFO: Length of Client Monitoring Information
- LOGID: Default Logical Database ID
- LUINFO: Length of User Data Passed to Adabas
- LUSAVE: Size of User Save Area
- LX1NAME: User Exit 1 Module Name
- LX2NAME: User Exit 2 Module Name
- MRO: Multiple Region Option
- NAZPDO: Natural zIIP Parameter Data Offset
- 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
- REVHID: Adabas Review Hub ID Support
- REVIEW: Adabas Review Support
- REVREL: Adabas Review Release
- RMI: Resource Manager Interface
- RST1BDB: Restore single-byte DBID for ACB calls
- RTXNAME: Command Retry Exit Name
- RVCLNT: Adabas Review Client Reporting Allowance Setting
- SAF: Adabas Security Interface Flag
- SAP: SAP Application Support
- SAPSTR: SAP ID String
- SVCNO: Adabas SVC number
- TCPIP: TCP/IP Support
- 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 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}

AUDIT: Adabas Auditing Support

Parameter	Description	Syntax
AUDIT	Indicates whether or not Adabas Auditing is to be activated for the link module(s) using this link globals table. ■ AUDIT=NO: Adabas Auditing will not be activated. ■ AUDIT=YES: Adabas Auditing will be activated.	AUDIT={NO|YES}

AVB: Adabas Bridge for VSAM Support

Parameter	Description	Syntax
AVB	Indicates whether or not 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: Adabas System Coordinator Support

Parameter	Description	Syntax
COR	Indicates whether or not the Adabas System Coordinator (COR) client component is to be activated for the link module(s) using this link globals table (as required by Adabas Fastpath, Adabas Vista, and Adabas Transaction Manager). ■ COR=NO: The COR client is *not* to be activated. ■ COR=YES: The COR client component is to be activated.	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 is used by a 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, read *Establishing Adabas SVC Routing by Adabas Database ID*, in the *Adabas z/OS Installation Guide* documentation. **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	DBSVCTN={*name*|ADASVCTB}

Parameter	Description	Syntax
	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.	

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, read *Establishing Adabas SVC Routing by Adabas Database ID*, in the *Adabas z/OS Installation Guide* documentation.	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*}

FIXLOGID: Prevent Runtime Modification of LOGID

Parameter	Description	Syntax
FIXLOGID	**Note:** This parameter is only appropriate for sites using Adabas Vista and must only be used with agreement from Software GmbH. This parameter prevents the runtime modification of the default database ID defined by the **LOGID** parameter. ■ FIXLOGID=NO: The default database ID may be modified at runtime. ■ FIXLOGID=YES: The default database ID may *not* be modified at runtime.	FIXLOGID={NO|YES}

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}

LMINFO: Length of Client Monitoring Information

Parameter	Description	Syntax
LMINFO	The length of the client monitoring information area for use by Adabas Auditing (AUDIT=YES) and/or Adabas Review (REVIEW=YES). If AUDIT=YES and/or REVIEW=YES is specified, a length of 640 should be specified for LMINFO. However, for backwards compatibility, if LMINFO=0 then a length of 640 will automatically be generated (accompanied by an MNOTE informational message) when this link globals table is assembled. Refer to the relevant Adabas Auditing and/or Adabas Review documentation for any additional information regarding the setting of this parameter. If both products are to be activated, then specify the larger of any documented lengths	LMINFO={0|*length*}

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}

LUINFO: Length of User Data Passed to Adabas

Parameter	Description	Syntax
LUINFO	The length of user data to be passed between the Adabas link module user exits 1 and 2 and Adabas nucleus user exit 11 and 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*}

Parameter	Description	Syntax
	For more information regarding the passing of user data refer to User Buffers in the chapter *Command Reference*.	

LUSAVE: Size of User Save Area

Parameter	Description	Syntax
LUSAVE	This parameter is redundant and will be dropped in a future version of Adabas. We strongly suggest to remove any use of this parameter from your LGBLSET input. On entry to the Adabas link module user exits 1 and 2, register 2 will point to an 18 fullword area for the saving of registers. For more information regarding Adabas link module user exits 1 and 2, refer to **Link Routine User Exit 1 (Pre-Command) and User Exit 2 (Post-Command)** in the **Preparing Adabas Link Routines** section of the Installation for z/OS chapter.	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}

NAZPDO: Natural zIIP Parameter Data Offset

Parameter	Description	Syntax
NAZPDO	**Note:** This parameter is only appropriate for sites using LGBLSET USERX1=YES with applications running the zIIP Enabler for Natural and Natural Profile parameter ADAPRM=ON A NAZPDO value of zero (the default), results in no changes to the existing user exit 1 (and, if present, user exit 2) processing. That is, the presence of user exit 1 will result in SRB mode Adabas calls being switched to TCB mode before the exit is invoked. A NAZPDO value of non-zero (minimum value 2) results in the bypass of user exit 1 (and, if present, user exit 2) enabling Adabas calls to remain in SRB mode. In order to mitigate the bypassing of user exit 1, Natural's ADAPRM=ON data is passed through to the Adabas target as part of the user information area of the Adabas call. The NAZPDO value specified defines an offset from the beginning of the user information area (UBUINFO) which identifies the start of an area of length 260 bytes available for use by the link module for the passing of Natural's ADAPRM=ON data. The format of this 260 byte area is a 4 byte eye-catcher "NZP1" followed by 256 bytes of Natural's ADAPRM=ON data. **Notes:** 1. The length of the user information area specified by the LGBLSET LUINFO= parameter must be a minimum value of NAZPDO_offset+260. 2. In conjunction with the use of this parameter the LU and NAB parameters of the relevant Adabas targets should be reviewed and increased as appropriate. This is due to the increase in the length of the user information area when this parameter is in use. For more information on the use of this parameter contact your Adabas technical support representative.	NAZPDO={0|offset}

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 "C" plus the CICS task number, in zoned decimal format, including leading zeroes. 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	NETOPT={NO|YES}

Parameter	Description	Syntax
	assembler MNOTE and a return code of 16 are produced from the assembly step.	

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 SP8 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 SP8), 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}

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 an Adabas ACBX direct call. This is because the TWA is of finite size per transaction and because the TWA is not available at CICS startup. We therefore recommend that CICS programs 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}

REVHID: Adabas Review Hub ID Support

Parameter	Description	Syntax
REVHID	Specifies the preferred Adabas Review hub ID. This value can be checked during the Adabas TP monitoring installation or during the monitor activate process. If REVHID is set to zero (0), the preferred Adabas Review hub ID is dynamic. When the hub ID is dynamic, it cannot be checked during the Adabas TP monitoring installation and the call to turn on client reporting must supply the correct Adabas Review hub ID to use. If REVHID is specified, REVIEW=YES must also be specified. If REVHID is specified and REVIEW=NO is also specified, the assembly of the globals table will abort with condition code 16 and the following message is given: REVHID requires REVIEW=YES	REVHID=*hubid*

REVIEW: Adabas Review Support

Parameter	Description	Syntax
REVIEW	Indicates whether or not Adabas Review performance monitor is to be activated for the link module(s) using this link globals table. REVIEW=NO will not activate the Adabas Review performance monitor. REVIEW=YES will automatically activate the Adabas Review performance monitor for all clients. REVIEW=COR requires COR=YES and is only applicable to clients running with the Adabas System Coordinator. REVIEW=COR indicates that client activation of the Adabas Review performance monitor will be deferred to the setting of the Adabas System Coordinator's client runtime control "Review". Refer to the Adabas System Coordinator documentation for more information on this client runtime control.	REVIEW={NO|YES|COR}

REVREL: Adabas Review Release

Parameter	Description	Syntax
REVREL	This parameter is redundant and will be dropped in a future Adabas version. Please remove any use of this parameter from your LGBLSET input. Continued use of this parameter will result in the following informational MNOTE message: `REVREL= is redundant and is no longer required.` The assembly of the globals table is unaffected.	REVREL={}

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 and use of the CICS Resource Manager Interface.	RMI={NO|YES}

RST1BDB: Restore single-byte DBID for ACB calls

Parameter	Description	Syntax
RST1BDB	**Note:** This parameter is only appropriate for sites using Adabas Vista and must only be used with agreement from Software GmbH. For ACB calls using a single-byte DBID, this parameter determines if the DBID supplied in the first byte of the ACBFNR field is restored before returning the completed call to the application If RST1BDB is NO (the default), the DBID will not be restored. If RST1BDB is YES, the DBID will be restored.	RST1BDB={NO|YES}

RTXNAME: Command Retry Exit Name

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

RVCLNT: Adabas Review Client Reporting Allowance Setting

Parameter	Description	Syntax
RVCLNT	Indicates whether Adabas Review client reporting should be allowed and available for activation. RVCLNT=YES requires REVIEW=YES and indicates Adabas Review client reporting is allowed and available for the manual activation of all clients. Refer to the Adabas Review documentation for the means of activation. RVCLNT=COR requires REVIEW=YES or COR and is only applicable to clients running with the Adabas System Coordinator. RVCLNT=COR indicates Adabas Review client reporting is allowed and activation is deferred to the setting of the Adabas System Coordinator's client runtime control 'Client Monitor'. Refer to the Adabas System Coordinator documentation for more information on this client runtime control.	RVCLNT={NO|YES|COR}

SAF: Adabas Security Interface Flag

Parameter	Description	Syntax
SAF	Indicates whether Software GmbH's Adabas SAF Security support is required. In CICS environments only, if you want your security system user IDs to be stored in Adabas user queue elements (making them available for display and review as well as preventing response code 200, ADARSP200, subcode 21 when ADARUN SECUID=REQUIRE is in effect for Adabas), you must code the SAF parameter as YES. This is only required in CICS environments; in other environments, the security system user IDs are automatically stored.	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. Valid values range from 200-255 and the default is "249".	SVCNO=*nnn*

TCPIP: TCP/IP Support

Parameter	Description	Syntax
TCPIP	Indicates whether communication using TCP/IP sockets is supported. If TCPIP=YES then communication to any database registered in the Adabas Directory Server will be performed using TCP/IP sockets.	TCPIP={NO|YES}

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. **Caution:** Be sure to specify a TP monitor operating environment that is supported on the operating system you selected in the OPSYS parameter.	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 GmbH 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	UBTYPE={POOL|TASK}

Parameter	Description	Syntax
	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. UBTYPE=POOL should be used if any of the following are true: ■ The majority of Adabas CICS transactions are short running tasks issuing a relatively small number of Adabas calls per CICS task. ■ The CICS system is subject to CICS storage fragmentation. ■ Applications running the Natural for zIIP are present in this CICS system. Otherwise, UBTYPE=TASK may be considered if: ■ The majority of Adabas CICS transactions are long running tasks issuing a relatively large number of Adabas calls per CICS task. ■ CICS tasks frequently trip CPU limits set by CICS execution monitoring programs such as those from Omegamon.	

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 task-related user exit (TRUE). XWAIT=YES is the default. The CICS WAIT EVENTS EXTERNAL (XWAIT=YES) is the recommended interface for all supported versions of CICS/TS. The CICS WAITCICS statement (XWAIT=NO) is provided but may result in poor CICS transaction performance or unpredictable transaction results in busy CICS environments.	XWAIT={NO|YES}

Notes:

1. If XWAIT=NO is specified, the ADACICT (Adabas 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.
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.

9 Installing Adabas Add-ons (including zIIP)

- License 132
- Using the Adabas System Coordinator-based Add-ons 133

The installation of the following add-on products can be done alongside an Adabas installation as the Adabas installation package includes a set of sub-products or add-ons:

- Adabas for zIIP (AZP)
- Adabas Caching Facility (ACF)
- Adabas Delta Save Facility (ADE)
- Adabas Cluster Services (ALS)
- Adabas Parallel Services (ASM)
- Adabas Online System (AOS)

In order to be able to use the sub-products they must be licensed and the corresponding product license file must be provided with the nucleus or utility execution.

This chapter covers the following topics:

License

An Adabas nucleus that is to run with the Adabas for zIIP, Adabas Caching Facility, Adabas Delta Save Facility, Adabas Cluster Services, Adabas Parallel Services or Adabas Online System add-on product requires an associated license file in addition to the Adabas license file. If the license is not provided or erroneous, the nucleus will run with the add-on deactivated.

An add-on license can be transferred to the mainframe in the same way as the Adabas license. The license can be made available to the nucleus as a load module with the name DDL*ccccc*, where *ccccc* represents the product code. This is similar to the handling of the Adabas license load module ADALIC described in ***Step 5: Customize and Start the Adabas Nucleus and Test Adabas Communications***, elsewhere in this guide.

Alternatively, the license file can be referred to by a 'DDL*ccccc*' (where *ccccc* represents the product code) DD statement in the nucleus job/started task. This is a fallback in case the DDL*ccccc* module cannot be loaded.

For a list of license file names, load modules and DD/link names, refer to ***Adabas and Add-on Licenses***, elsewhere in this guide.

Using the Adabas System Coordinator-based Add-ons

If Adabas Vista, Adabas Fastpath, Adabas Transaction Manager, or Adabas SAF Security is being used, please refer to Using COR-based Add-ons in the Release Notes for installation instructions.

10 Enabling Universal Encoding Support (UES) for Your Adabas Nucleus

Adabas is delivered with its own data conversion capability called *universal encoding support (UES)*. Entire Net-Work detects when it is connected to a target database that converts data and passes the data through to Adabas without converting it.

To ensure UES processing is handled properly, perform the following steps.

1. The Adabas database must include the correct libraries and have appropriate zaps applied.

   A sample startup job for a UES-enabled nucleus is provided in member ADANUCU of the ADA`vrs`.JOBS data set. For more information, read *JCL Required for UES and TCP/IP Support (z/OS)*.

   In addition, read *Universal Encoding Support (UES)* in *Adabas DBA Tasks Manual* as well as *ADADEF Utility: Define a Database* and *ADACMP Utility: Compress-Decompress Data* in *Adabas Utilities Manual* for more information.

2. Verify that the load modules for all Adabas 8 link routines have been linked with the default (or updated) translation tables and that the LGBLSET **SVCNO parameter** has been set. For information on altering UES enablement in the link routines read appropriate sections of ***Installing Adabas With TP Monitors***, elsewhere in this guide, starting with the section ***UES-Enabled Link Routines***.

   UES-enabled databases can be connected to machines with different architectures through Com-plete, Entire Net-Work (WCP), and optionally in a z/OS environment through a direct TCP/IP link to the Adabas nucleus from web-based applications or from PC-based applications such as Software GmbH's Jadabas. Connections through Com-plete use the Adabas Com-plete link routines; connections through Entire Net-Work use the Adabas batch link routines. Connections through a direct TCP/IP link are described in the remainder of this chapter.

Note: The use of UES-enabled link routines and a UES-enabled nucleus is transparent to applications, including applications that do not require universal encoding translation support. Therefore, it is not necessary to disable UES if it is already enabled.

11 Enabling Direct TCP/IP Access (ADATCP) to Your Adabas Nucleus

ADATCP runs as a subtask in the Adabas address space. ADATCP utilizes the Adabas Directory Server (ADI) by registering its target ID. This entry contains the target ID, host name, and port number, providing the information the application needs to send Adabas commands directly to a remote Adabas nucleus.

ADATCP also registers its kernel with the ADI, so that Adabas Manager (AMN) can communicate with it. At session end, ADATCP deletes its entries from the ADI.

A database must be enabled for UES to run with ADATCP. ADATCP requires a valid Net-Work (WCP) license, in addition to any other Adabas licenses.

For more information on Direct TCP/IP Access (ADATCP) see the documentation for *Entire Net-Work Mainframe*, section *Point-to-Point Support for Adabas (ADATCP)*.

12 Installing The AOS Demo Version

- AOS Demo Installation Procedure 140
- Installing AOS with Natural Security 141
- Setting the AOS Demo Version Defaults 142

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

Notes:

1. To install the full version of AOS, see the *Adabas Online System* documentation.
2. Demo versions of Adabas Vista (AVI), Adabas Fastpath (AFP), Adabas SAF Security (AAF), and Adabas Transaction Manager (ATM) are automatically installed when you install either the demo or full version of AOS. If you are using the full version of any of the above products, use the SYSPROD Natural system command on each full version product to check if you have installed the latest cumulative INPL update from Empower. Install any missing updates as described in the specific product's documentation.

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

AOS Demo Installation Procedure

To install the AOS demo version without the System Maintenance Aid

1 For a Com-plete or CICS environment, link the correct object module with the Natural TP nucleus.

If a split Natural nucleus is to be installed, the AOSASM module must be linked to the shared portion of the nucleus and not to the thread portion.

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

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

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

4 Perform a Natural INPL.

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

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

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

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

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

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

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

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

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

Installing AOS with Natural Security

If Natural Security is installed, define at least the library SYSAOS to it.

Define the following libraries as needed:

- For Adabas Vista: SYSAVI and SYSMV*vrs*
- For Adabas Fastpath: SYSAFP and SYSMW*vrs*

- For Adabas SAF Security: SYSAAF and SYSMX*vrs*
- For Adabas Transaction Manager: SYSATM and SYSMT*vrs*

we recommend you define SYSAOS and any other libraries you may define as protected.

Specify the startup program for SYSAOS as MENU. Do not specify a startup program name for the other libraries.

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

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

Setting the AOS Demo Version Defaults

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

The table below lists the parameters and possible values.

Parameter	Valid Values	Default	Description
ADMIN-LEVEL	0-9	6	Administration level: Allows access to certain functions that can cause error conditions in the ADABAS environment. When set to 8 or higher, it allows the "CATCH RSP-CODE" direct command to occur, and when set to 9, it allows the "ZAP" function to be issued.
AOS-END-MSG	Yes (Y) or No (N)	Y	Display AOS end-of-session message?
AOS-LOGO	Yes (Y) or No (N)	N	Display AOS logo?
BATCH-ERROR	Yes (Y) or No (N)	N	Batch job cond code: When AOS is executing from a batch job and has an error condition, and if BATCH-ERROR is set to "Y", AOS will terminate with a condition code of 8. This function will be fully implemented over time, as all AOS programs must be modified for this.
BLK-CYL	Cylinder (C) or Block (B)	B	Space control by block or cylinder

Parameter	Valid Values	Default	Description
CMD-INT	Natural (N) or AOS (A)	A	Pass-through control for invalid AOS commands: "N" passes invalid commands to Natural; "A" displays an error message for invalid commands.
CPEXLIST	No (N) or Yes (Y)	N	Display extended checkpoint list? A value of "N" displays the normal list; a value of "Y" displays the extended list.
EX1-A1	---	---	Reserved for future use.
EX1-N3	---	---	Reserved for future use.
EXF-UTI	E or U	U	UTI or EXF file lock exception. A value of E specifies an EXF exception; a value of U specifies a UTI exception.
HWM-PRT	■ ' ' **(blank)** The selection fields are not displayed, with lines having 9 digit counters/multiplier. ■ **L** The selection fields are not displayed, with lines having full 26 digit counters, and extra screens. ■ **N** The selection fields are displayed, with lines having 9 digit counters/multiplier.	' ' (blank)	Batch report function: HWM-PRT is an option for displaying/printing the selection fields on each line of the High Watermarks for SYSAOS Batch execution jobs. There is also a value for complete counter displays.
MAX-AC-IOS	0-999999	150	AC read converter block threshold value
MAXANZ	1-99999999	100	Maximum displayed user queue elements
NR-EXT	1-5	4	Critical extent threshold for listing file. This parameter applies to Adabas 7.4 (or earlier) installations.
NR-EXT2	1-99	50	Critical extent threshold for listing file. This parameter applies to Adabas 8 (or later) installations.
NR-PERCENT	1-99	89	Report function: NR-PERCENT is a threshold value for the display of critical files concerning the percentage full of the extents reached in AC/UI/NI/DS table type. A value greater than NR-PERCENT will be highlighted.
PURGE-UQE	Yes (Y) or No (N)	N	Remove user queue element?

Parameter	Valid Values	Default	Description
SAVEFDT	Yes (Y) or No (N)	N	Keep deleted file's FDT?
STATINTV	1-9999 seconds	60	Statistics-gathering interval, in seconds
TID-DISPLAY	B, A, I	I	Control display for TID in "display user queue" function: "B" = binary TID display; "A" = alpha TID display; "I" = alpha for A-Z/0-9, otherwise binary.
TIMELA	0-99999999 seconds	0 (no limitations)	Display user queue elements with activity during the last "n" seconds.
TIN-JOBN	T or J	J	Display either job name or time-in in "display command queue" function. A value of "T" indicates that time-in should be displayed; a value of "J" indicates that the job name should be displayed.

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

```
  .
    .
    .
DEFINE DATA PARAMETER USING P-AOSEX1
END-DEFINE
*
* SET THE DEFAULTS
*
ADMIN-LEVEL = '6'    (Allows access to certain functions that can cause error ↩
conditions in the ADABAS environment)
AOS-END-MSG = 'Y'    (Display end-of-session message)
AOS-LOGO = 'Y'       (Adabas Online System logo display-set to 'N' for no logo ↩
display)
BATCH-ERROR = 'N'    (If BATCH-ERROR is set to "Y", AOS will terminate with a ↩
condition code of 8 if an error occurs.)
BLK-CYL  = 'B'       (Space allocation default-set to 'C' for cylinders)
CMD-INT  =  'A'      (Pass invalid Adabas commands to (N)atural, or intercept (A))
CPEXLIST = 'N'      (Checkpoint list control-set to 'Y' for extended checkpoint list)
NR-EXT2 = '50'       (ADA V8 critical extent threshold.  Range: 1-99)
EXF-UTI  = 'U'      (File locking exception-set to 'E' to except files in EXF status)
HWM-PRT =  ' '      (Print/display option for high-water mark counters: blank, 'L' ↩
or 'N')
MAXANZ  = 100       (Maximum user queue elements displayed. range: 1 - 99,999,999 ↩
elements)
NR-EXT = 4           (ADA V7 critical extent threshold.  Range: 1, 2, 3, 4, or 5)
NR-PERCENT = '89'    (NR-PERCENT is a threshold value for the display of critical ↩
files)
MAX-AC-IOS = 150     (AC read converter block threshold)
PURGE-UQE  = 'N'     (Remove element from user queue. Pre-5.1 default is "Y")
SAVEFDT  = 'N'       (Keep old FDT for SAVE operation-set to 'Y' to save FDTs)
```

```
STATINTV = 60        (Statistic-gathering time. range: 1 - 9999)
TID-DISPLAY = 'I'    (TID display control: B=binary, A=alpha, I=normally alpha, ↩
special characters as binary)
TIMELA = 0          (Include activity in last 'n' seconds. range: "all" (0) -last ↩
99,999,999 seconds)
TIN-JOBN = 'J'       (Command queue display-"J" for job name, "T" for "time in ↩
queue" )
*
END
```

13 Installing the Recovery Aid (ADARAI)

- ADARAI Installation Overview 148
- ADARAI Installation Procedure 148

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

ADARAI Installation Overview

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

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

ADARAI Installation Procedure

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

To install the Adabas Recovery Aid:

1. Define and format the DDRLOGR1 file.

   Use the ADAFRM RLOGFRM function to format the RLOG.

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

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

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

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

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

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

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

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

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

3 Install ADARAI on the database.

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

4 Create the first ADARAI generation.

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

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

14 Installing The Error Handling And Message Buffering Feature

This section describes how to install the error handling and message buffering feature.

To install the error handling and message buffering feature:

1 Specify ADARUN `SMGT=YES`. If message buffering is to be used, also specify ADARUN `MSGBUF` with a value greater than zero.

When ADARUN `SMGT=YES` is specified to activate the error handling tool, the initialization module ADAMXI is loaded by ADARUN and is then called during session open:

- the error handling header/environment is initialized;
- the message buffer is initialized if ADARUN `MSGBUF` is specified with a value greater than zero;
- the error handling modules are loaded into memory by ADAIOR;
- the Adabas module table is built;
- any provided error handling user exit is initialized;
- the default recovery plug-in (PIN) module ADAMXY is installed;
- the response code analyzer plug-in (PIN) module PINRSP is installed;
- the program check and abnormal termination handlers are activated;
- the error handling flag in the header is raised indicating a successful start;
- the ADANI2 message is generated to indicate that error handling is active in the nucleus.

2 Decide which exits are critical (the default) and issue `SMGT,XNOTCRITICAL=`*`exit-code`* operator commands for those that are not critical.

3 Customize ADASMXIT if necessary, particularly if PINRSP or PINAUTOR are to be activated or output is to be directed to DDTRACE1 rather than DDPRINT for the PIN routines. Reassemble the exit and ensure that it resides in the Adabas load library or is available in a load library that is available at startup time.

4 Decide which PINs to activate.

The following table lists the available PINs and how to activate them:

PIN Routine	To install ...
PINAUTOR	rename NOAUTOR in the Adabas load library to PINAUTOR
PINOPRSP	rename NOOPRSP in the Adabas load library to PINOPRSP
PINUES	issue operator command `SMGT,ADDPIN=PINUES` when the nucleus is active (see note below)

Note: Since PINRSP and PINUES handle some of the same response codes, perform the ADDPIN function last on the module that is to acquire control. For example, PINRSP and PINUES both handle a response code 55 (ADARSP055). If PINUES is to acquire control, the ADDPIN must be done on PINUES after PINRSP.

At this point error handling is fully operational and `SMGT` operator commands may be issued.

15 Adabas Dump Formatting Tool (ADAFDP)

- ADAFDP Function 154
- ADAFDP Output 154

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

ADAFDP Function

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

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

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

ADAFDP Output

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

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

ADAFDP Messages

Message	Description
ADAH01 / ADAH02	The message is displayed on the console and written to DDPRINT at the point where the format begins and terminates.
ADAMPM ABEND CODE and PSW	If an Abend code and program status word (PSW) were saved in ADAMPM by the Adabas ESTAE, ADAFDP displays these. In addition, ADAFDP determines the module whose entry point best fits the PSW and calculates the offset within that module. If the ADAMPM abend code and PSW are zero, ADAFDP does not format this information.
ADABAS MODULE LOCATIONS	ADAFDP formats and displays the location of each of the Adabas nucleus modules resident in the address space.
ADDRESS LOCATIONS FOR USER EXITS	ADAFDP formats and displays the location of any user exit loaded with the Adabas nucleus.

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

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

Pool Abbreviations

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

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

User Threads

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

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

Command Information

Information	Description
CQE Address	The address location of this CQE
F	Command queue flag bytes: ■ First Byte: General Purpose Flag ■ X'80': User buffers in service partition, region, address space ■ X'40': ET command waiting for 12 call ■ X'20': Waiting for 16 call ■ X'10': 16 call required ■ X'08': Attached buffer ■ X'04': Attached buffer required ■ X'02': X-memory lock held (MVS only) ■ Second Byte: Selection Flag ■ X'80': In process ■ X'40': Ready to be selected ■ X'20': Search for UQE done ■ X'10': UQE found ■ X'08': Not selectable during BSS=x'80' status ■ X'04': Not selectable during ET-SYNC ■ X'02': Waiting for space ■ X'01': Waiting for ISN in HQ
CMD	The command type
File Number	The file number for this command
Job Name	Job name for the user
Addr User	UQE Address of users UQE, if searched for and found

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

RABN Information

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

16 Translation Tables

- Adabas EBCDIC to ASCII and ASCII to EBCDIC 162
- Entire Net-Work EBCDIC to ASCII and ASCII to EBCDIC 163

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

Adabas EBCDIC to ASCII and ASCII to EBCDIC

```
cUES2ASC DS 0F
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c DC x'000102033F093F7F3F3F3F0B0C0D0E0F' 0.
c DC x'101112133F3F083F18193F3F3F1D3F1F' 1.
c DC x'3F3F1C3F3F0A171B3F3F3F3F3F050607' 2.
c DC x'3F3F163F3F1E3F043F3F3F3F14153F1A' 3.
c DC x'203F3F3F3F3F3F3F3F3F3F2E3C282B3F' 4.
c DC x'263F3F3F3F3F3F3F3F3F21242A293B5E' 5.
c DC x'2D2F3F3F3F3F3F3F3F3F7C2C255F3E3F' 6.
c DC x'3F3F3F3F3F3F3F3F3F603A2340273D22' 7.
c DC x'3F6162636465666768693F3F3F3F3F3F' 8.
c DC x'3F6A6B6C6D6E6F7071723F3F3F3F3F3F' 9.
c DC x'3F7E737475767778797A3F3F3F5B3F3F' A.
c DC x'3F3F3F3F3F3F3F3F3F3F3F3F5D3F3F' B.
c DC x'7B4142434445464748493F3F3F3F3F3F' C.
c DC x'7D4A4B4C4D4E4F5051523F3F3F3F3F3F' D.
c DC x'5C3F535455565758595A3F3F3F3F3F3F' E.
c DC x'3031323334353637383939F3F3F3F3F3F' F.
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END
cUES2EBC DS 0F
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c DC x'00010203372D2E2F1605250B0C0D0E0F' 0.
c DC x'101112133C3D322618193F27221D351F' 1.
c DC x'405A7F7B5B6C507D4D5D5C4E6B604B61' 2.
c DC x'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F' 3.
c DC x'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' 4.
c DC x'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D' 5.
c DC x'79818283848586878889919293949596' 6.
c DC x'979899A2A3A4A5A6A7A8A9C06AD0A107' 7.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' 8.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' 9.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' A.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' B.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' C.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' D.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' E.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' F.
c* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END
```

Entire Net-Work EBCDIC to ASCII and ASCII to EBCDIC

```
NW2ASC DS 0F
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
DC X'000102030405060708090A0B0C0D0E0F' 0.
DC X'101112131415161718191A1B1C1D1E1F' 1.
DC X'00000000000000000000000000000000' 2.
DC X'00000000000000000000000000000000' 3.
DC X'20000000000000000000005B2E3C282B5D' 4.
DC X'26000000000000000000021242A293B5E' 5.
DC X'2D2F0000000000000007C2C255F3E3F' 6.
DC X'000000000000000000603A2340273D22' 7.
DC X'00616263646566676869000000000000' 8.
DC X'006A6B6C6D6E6F707172000000000000' 9.
DC X'007E737475767778797A00005B000000' A.
DC X'00000000000000000000000005D0000' B.
DC X'7B41424344454647484900000000000' C.
DC X'7D4A4B4C4D4E4F505152000000000000' D.
DC X'5C7E535455565758595A000000000000' E.
DC X'30313233343536373839700000000FF' F.
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
NW2EBC DS 0F
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
DC X'000102030405060708090A0B0C0D0E0F' 0.
DC X'101112131415161718191A1B1C1D1E1F' 1.
DC X'405A7F7B5B6C507D4D5D5C4E6B604B61' 2.
DC X'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F' 3.
DC X'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' 4.
DC X'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D' 5.
DC X'79818283848586878889919293949596' 6.
DC X'979899A2A3A4A5A6A7A8A9C06AD0A100' 7.
DC X'00000000000000000000000000000000' 8.
DC X'00000000000000000000000000000000' 9.
DC X'00000000000000000000000000000000' A.
DC X'00000000000000000000000000000000' B.
DC X'00000000000000000000000000000000' C.
DC X'00000000000000000000000000000000' D.
DC X'00000000000000000000000000000000' E.
DC X'00000000000000000000000000000FF' F.
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END
```

17 Installing Adabas Online Services for z/OSMF Utility Server

- AOZ Server Licensing Requirements 166
- AOZ Server Installation Steps 166

This document provides information on the Adabas Online Services for z/OSMF Utility Server with product code AOZ, which was first introduced with Adabas 8.5 SP4 in conjunction with Adabas Manager 9.2 SP0.

AOZ is a RESTful server that leverages z/OSMF framework of IBM to provide APIs for Adabas Manager (AMN) to enable Adabas utility job submission and job output display.

The following section describes only how to install the AOZ server. Refer to the Adabas Manager documentation for information on the administration features that require the installation of the AOZ server.

Note: The AOZ server is distributed with Adabas Online Services.

AOZ Server Licensing Requirements

For correct operation of the AOZ server, the Adabas Online Services license file must be available in the target Adabas servers, which are administered by Adabas Manager.

AOZ Server Installation Steps

The following table describes the steps needed to successfully install the AOZ server.

Step	Description
1	**Create a UNIX System Services installation directory** Create a directory in the UNIX System Services file system to use as your installation directory. For the purposes of these installation instructions, the chosen installation directory is /opt/softwareag. `$ mkdir -p /opt/softwareag`
2	**Copy the AOZ package to the installation directory** Copy the AOZ distribution package to the /opt/softwareag installation directory. `cp "//'hlq.AOZvrs.TAR'" /opt/softwareag/AOZvrs.tar.Z`

Step	Description

3	**Extract the AOZ package into the installation directory**
	
	Extract the contents of the AOZ package in the UNIX System Services installation directory.
	
	`$ cd /opt/softwareag`
	`$ uncompress < AOZvrs.tar.Z | tar -oxf -`
	
	The following components are extracted into /opt/softwareag/aoz:
	
	■ `aoz-server-<version>.jar` – AOZ server .jar file
	■ `AOZSTC` – Sample JCL to run the AOZ server as a started task

4	**Copy the sample JCL to the z/OS procedure library**
	
	Copy the sample JCL for the AOZ server started procedure to the z/OS procedure library.
	
	`$ cd /opt/softwareag/aoz`
	`$ cp AOZSTC "//'hlq.PROCLIB(AOZSTC)'"`

5	**Tailor the sample JCL for starting the AOZ server** Modify the following variables in the JCL for the started procedure: ■ `ZOSMF` – URL of z/OSMF ■ `JAVA` – Java installation directory ■ `AOZDIR` – Full path to the AOZ installation directory ■ `AOZVER` – Version number of the AOZ server .jar file ■ `AOZPORT` – Port number for the AOZ server to listen

Index

, 125

A

ACINAMES module, 67, 77
ACIOPT table, 67
Adabas
  CICS execution unit, 66
  installation for z/OS, v
Adabas Bridge for DL/I support, 115
Adabas Bridge for VSAM support, 115
Adabas CICS task-related user exit (TRUE)
  module name, 127
Adabas Online System (AOS)
  AOSEX1 program parameters, 142
  modify default parameter values, 142
  setting defaults, 142
Adabas Review hub ID support, 123
Adabas Review release, 124-125
Adabas Review support, 124
Adabas security interface parameter, 126
Adabas SVC number parameter, 126
Adabas System Coordinator Support, 116
ADACICS module, 66, 78
ADACICT module, 76
ADADCI module, 66
ADATCP support
  enabling in z/OS, 137
ADL parameter, 115
ADMIN-LEVEL parameter, 142
AOS-END-MSG parameter, 142
AOS-LOGO parameter, 142
AOSEX1 user exit, 142
  setting defaults, 142
AUDIT parameter, 115
AVB parameter, 115

B

BATCH-ERROR parameter, 142
BLS-CYL parameter, 142

C

CICS application stub, 66, 78
CICS command-level link routine name, 117
CICS execution unit, 66
CICS installation options table, ACIOPT, 67, 75
CICS multiple region option, 119
CICS names module, ACINAMES, 67, 77
CICS purge transaction parameter, 123
CICS Resource Manager Interface parameter, 124
CICS user ID creation method, 120
CICS XWAIT setting, 130
CITSNM parameter, 116
CMD-INT parameter, 143
CNAME parameter, 69
command retry exit name, 125
COR parameter, 116
CPEXLIST parameter, 143
CSECT or DSECT generation, 118

D

DBID parameter, 111
DBID/SVC routing table, 116-117
  source code, 109
DBID2 parameter, 112
DBSVCTN parameter, 116
default target database ID, 118
defaults, 142
DSECT data prefix parameter, 122
DYNDBSVC parameter, 117

E

ENTPT parameter, 79, 117
ENTRY=FINAL statement, 71
ENTRY=GLOBAL statement, 69
ENTRY=GROUP statement, 71
EX1-A1 parameter, 143
EX1-N3 parameter, 143
EXF-UTI parameter, 143

F

files
  large format, 32
FIXLOGID parameter, 117

G

GBLNAME parameter, 118
GEN parameter, 69, 118
GTNAME parameter, 68, 71

H

HWM-PRT parameter, 143

I

IMQNAME parameter, 70
IMSGDEST parameter, 70
installation
  for z/OS, v
Installing Base Adabas Manually, 41

L

large format files, 32
Length of Client Monitoring Information, 118
length of user data passed to user exit 4, 118
LGBLSET macro
  ADL parameter, 115
  AUDIT parameter, 115
  AVB parameter, 115
  CITSNM parameter, 116
  COR parameter, 116
  DBSVCTN parameter, 116
  DYNDBSVC parameter, 117
  ENTPT parameter, 117
  FIXLOGID parameter, 117
  GBLNAME parameter, 118
  GEN parameter, 118
  LMINFO parameter, 118
  LOGID parameter, 118
  LUINFO parameter, 118
  LUSAVE parameter, 119
  LX1NAME parameter, 119
  LX2NAME parameter, 119
  modifying, 113
  MRO parameter, 119
  NAZPDO parameter, 120
  NETOPT parameter, 120
  NTGPID parameter, 121
  NUBS parameter, 121
  OPSYS parameter, 122
  PARMTYP parameter, 122
  PRE parameter, 122
  PURGE parameter, 123
  RENT parameter, 123
  RETRYX parameter, 123
  REVHID parameter, 123
  REVIEW parameter, 124
  REVREL parameter, 124
  RMI parameter, 124-125
  RTXNAME parameter, 125
  RVCLNT parameter, 125
  SAF parameter, 126
  SAP parameter, 126
  SAPSTR parameter, 126
  SVCNO parameter, 126
  TPMON parameter, 127
  TRUENM parameter, 127
  UBPLOC parameter, 127
  UBSTIME parameter, 128
  UBTYPE parameter, 128
  UES parameter, 129
  USERX1 parameter, 129
  USERX2 parameter, 129
  XWAIT parameter, 130
link globals module name, 118
link globals table, 78
LMINFO parameter, 118
LOGID parameter, 118
LUINFO parameter, 118
LUSAVE parameter, 119
LX1NAME parameter, 119
LX2NAME parameter, 119

M

MACINS macro
  description, 67
  example, 68
  syntax, 67
MACIOPT macro
  ENTRY=FINAL statement, 71
  ENTRY=GLOBAL statement, 69
  ENTRY=GROUP statement, 71
  example, 71
  syntax, 68
macros
  MACINS, 67
MAX-AC-IOS parameter, 143
MAXANZ parameter, 143
MDBSVC macro
  parameters, 111
  statement types, 110
  TYPE=FINAL statement syntax, 111
  TYPE=GEN statement syntax, 111
  TYPE=INIT statement syntax, 110
  using, 109
MNTRUE parameter, 70
MRO parameter, 119
multiple CICS TRUE support
  overview, 66

N

Natural group ID, 121
NAZPDO parameter, 120
NETOPT parameter, 120
NR-EXT parameter, 143
NR-PERCENT parameter, 143
NR1-N3 parameter, 143
NTGPID parameter, 121
NUBS parameter, 121

O

operating system parameter, 122
OPSYS parameter, 112, 122

P

parameter list area, 122
PARMTYP parameter, 122
PRE parameter, 122
PREFIX parameter, 112
PURGE parameter, 123
PURGE-UQE parameter, 143

R

reentrant globals module flag, 123
RENT parameter, 123
retry command exit flag, 123
RETRYX parameter, 123
REVHID parameter, 123
REVIEW parameter, 124
REVREL parameter, 124
RMI parameter, 124
routing Adabas calls, 105
RST1BDB parameter, 125
RTXNAME parameter, 125
RVCLNT parameter, 125

S

SAF parameter, 126
SAP ID string parameter, 126
SAP parameter, 126
SAP user ID generation support parameter, 126
SAPSTR parameter, 126
SAVEFDT parameter, 144
setting AOS defaults, 142
STATINTV parameter, 144
SVC parameter, 112
SVC routing
    by database ID, 105
SVCNO parameter, 126

T

TABNAME parameter, 113
target database ID
    default, 118
task-related user exit (TRUE), 76
TCP/IP access (ADATCP)
    enabling in z/OS, 137
TID-DISPLAY parameter, 144
TIMELA parameter, 144
TIN-JOBN parameter, 144
TP monitors
    CICS application stub, 66
    CICS execution unit, 66
    CICS installation options table, ACIOPT, 67
    CICS names module, ACINAMES, 67
    MACINS macro, 67
TP operating environment parameter, 127
TPMON parameter, 127
TRUENAME parameter, 68
TRUENM parameter, 76, 79, 127
TYPE=FINAL statement
    MDBSVC macro, 110
    syntax, 111
TYPE=GEN statement
    MDBSVC macro, 110
    syntax, 111
TYPE=INIT statement
    MDBSVC macro, 110
    syntax, 110

U

UBPLOC parameter, 127
UBSTIME parameter, 128
UBTYPE parameter, 128
UES (see Universal Encoding Support (UES))
UES parameter, 129
Universal Encoding Support (UES)
    enabling in z/OS, 135
universal encoding support parameter, 129
user block
    pool allocation parameter, 127
    scan time parameter, 128
    type parameter, 128
user blocks created by CICS link routine, 121
user exit 1 flag, 129
user exit 1 module name, 119
user exit 2 flag, 129
user exit 2 module name, 119
user exit 4
    length of user data passed, 118
user exits
    AOSEX1, 142
USERX1 parameter, 129
USERX2 parameter, 129

X

XWAIT parameter, 130

Z

z/OS
    enabling direct TCP/IP (ADATCP) access, 137
    enabling UES support, 135