

# **Adabas**

## **Installation for z/VSE**

Version 8.4.2

October 2017

This document applies to Adabas Version 8.4.2 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 © 1971-2017 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. 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 AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://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 AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://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 AG.

**Document ID: ADAMF-VSE-INSTALL-842-20210929**

## Table of Contents

|   |     |
|---|-----|
| 1 About this Documentation .....  | 1   |
| Document Conventions .....  | 2   |
| Online Information and Support .....                                      | 2   |
| Data Protection .....   | 3   |
| 2 Installation for z/VSE .....  | 5   |
| 3 Supported Environments .....  | 7   |
| 4 Installation Procedure .....  | 9   |
| Installation Checklist .....  | 10  |
| Contents of the Release Tape .....  | 11  |
| Preparing to Install Adabas .....   | 12  |
| Initializing the Adabas Communication Environment .....                   | 17  |
| Installing the Adabas Database .....                                      | 23  |
| Migrating an Existing Database .....                                      | 39  |
| Logical Unit Requirements .....   | 39  |
| Job Exit Utility .....  | 40  |
| Acquiring Storage for the ID Table .....                                  | 44  |
| Acquiring Storage for the IIBS Table .....                                | 44  |
| SVC Work Areas .....  | 45  |
| Displaying Storage Allocation Totals .....                                | 45  |
| Calls from Other Partitions .....   | 45  |
| Dummy Sequential Files .....  | 45  |
| Backward Processing of Tapes and Cartridges .....                         | 46  |
| Applying Zaps (Fixes) .....   | 46  |
| Adabas 8 Adalink Considerations .....                                     | 52  |
| Setting Defaults in ADARUN .....  | 55  |
| 5 Installing Adabas with TP Monitors .....                                | 57  |
| Preparing Adabas Link Routines for z/VSE .....                            | 58  |
| General Considerations for Installing Adabas with CICS .....              | 62  |
| Installing Adabas with CICS under Adabas 8 .....                          | 64  |
| Installing the CICS High-Performance Stub Routine for Adabas 8 .....      | 79  |
| Installing Adabas with Com-plete under Adabas 8 .....                     | 95  |
| Installing Adabas with Batch under Adabas 8 .....                         | 97  |
| Establishing Adabas SVC Routing by Adabas Database ID .....               | 99  |
| Modifying Source Member Defaults (LGBLSET Macro) in Version 8 .....       | 108 |
| 6 Enabling Universal Encoding Support (UES) for Your Adabas Nucleus ..... | 123 |
| Installing UES Support for the Adabas Nucleus .....                       | 125 |
| 7 Enabling Direct TCP/IP Access (ADATCP) to Your Adabas Nucleus .....     | 129 |
| Installing TCP/IP Support for the Adabas Nucleus .....                    | 130 |
| 8 Device and File Considerations .....                                    | 135 |
| Supported z/VSE Device Types .....  | 136 |
| FBA Devices .....   | 137 |
| ECKD Devices .....  | 138 |
| Adding New Devices .....  | 138 |

|   |     |
|---|-----|
| User ZAPs to Change Logical Units .....                   | 142 |
| 9 Installing The AOS Demo Version .....                   | 147 |
| AOS Demo Installation Procedure .....                     | 148 |
| Installing AOS with Natural Security .....                | 149 |
| Setting the AOS Demo Version Defaults .....               | 150 |
| 10 Installing The Recovery Aid (ADARAI) .....             | 153 |
| ADARAI Installation Overview .....                        | 154 |
| ADARAI Installation Procedure .....                       | 154 |
| 11 Adabas Dump Formatting Tool (ADAFDP) .....             | 157 |
| ADAFDP Function .....                                     | 158 |
| ADAFDP Output .....                                       | 158 |
| 12 Maintaining A Separate Test Environment .....          | 165 |
| 13 Translation Tables .....                               | 169 |
| Adabas EBCDIC to ASCII and ASCII to EBCDIC .....          | 170 |
| Entire Net-Work EBCDIC to ASCII and ASCII to EBCDIC ..... | 171 |
| Index .....   | 173 |

# 1

## About this Documentation

---

|  |   |
|--|---|
| ■ Document Conventions .....           | 2 |
| ■ Online Information and Support ..... | 2 |
| ■ Data Protection .....                | 3 |

## Document Conventions

---

| Convention     | Description  |
|----------------|--|
| <b>Bold</b>    | Identifies elements on a screen.   |
| Monospace font | Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.  |
| <i>Italic</i>  | Identifies:<br><br>Variables for which you must supply values specific to your own situation or environment.<br>New terms the first time they occur in the text.<br>References to other documentation sources. |
| Monospace font | Identifies:<br><br>Text you must type in.<br>Messages displayed by the system.<br>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

---

### Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <https://documentation.softwareag.com>.

### Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to [empower@softwareag.com](mailto:empower@softwareag.com) with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at [https://empower.softwareag.com/public\\_directory.aspx](https://empower.softwareag.com/public_directory.aspx) and give us a call.

### **Software AG Tech Community**

You can find documentation and other technical information on the Software AG Tech Community website at <https://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have Tech Community credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

## **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 Installation for z/VSE

---

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

*Supported Environments*

*Installation Procedure*

*Installing Adabas with TP Monitors*

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

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

*Device and File Considerations*

*Installing the AOS Demo Version*

*Installing the Recovery Aid (ADARAI)*

*Adabas Dump Formatting Tool (ADAFDP)*

*Maintaining a Separate Test Environment*

*Translation Tables*

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*.



## 3 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 general information regarding Software AG product compatibility with other platforms and their requirements for Software AG products, visit Software AG's [Hardware Supported](#) web page; for specific information regarding Software AG product compatibility with IBM platforms and any IBM requirements for Software AG products, visit Software AG's [Product Compatibility for IBM Platforms](#) web page.



# 4

## Installation Procedure

---

|   |    |
|---|----|
| ■ Installation Checklist .....                            | 10 |
| ■ Contents of the Release Tape .....                      | 11 |
| ■ Preparing to Install Adabas .....                       | 12 |
| ■ Initializing the Adabas Communication Environment ..... | 17 |
| ■ Installing the Adabas Database .....                    | 23 |
| ■ Migrating an Existing Database .....                    | 39 |
| ■ Logical Unit Requirements .....                         | 39 |
| ■ Job Exit Utility .....                                  | 40 |
| ■ Acquiring Storage for the ID Table .....                | 44 |
| ■ Acquiring Storage for the IIBS Table .....              | 44 |
| ■ SVC Work Areas .....                                    | 45 |
| ■ Displaying Storage Allocation Totals .....              | 45 |
| ■ Calls from Other Partitions .....                       | 45 |
| ■ Dummy Sequential Files .....                            | 45 |
| ■ Backward Processing of Tapes and Cartridges .....       | 46 |
| ■ Applying Zaps (Fixes) .....                             | 46 |
| ■ Adabas 8 Adalink Considerations .....                   | 52 |
| ■ Setting Defaults in ADARUN .....                        | 55 |

This section describes the procedure for Adabas installation in z/VSE environments.

## Installation Checklist

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

| Step | Description   | Additional Information  |
|------|---|---|
| 1    | Allocate DASD space for the Adabas libraries.   | The libraries are restored from the installation media. Refer to the section <a href="#">Disk Space Requirements for Libraries</a> .                                      |
| 2    | Allocate DASD space for the Adabas database.  | For better performance, distribute the database files over multiple devices and channels. Refer to the section <a href="#">Disk Space Requirements for the Database</a> . |
| 3    | Specify a z/VSE partition for running the Adabas nucleus.   | Refer to the section <a href="#">Adabas Nucleus Partition/Address Space Requirements</a> .  |
| 4    | Define the library before restoration.  | See section <a href="#">Defining the Library</a> .  |
| 5    | Restore the Adabas libraries.   | See section <a href="#">Installing the Adabas Release Tape</a> .  |
| 6    | Install the Adabas SVC using the ADASIP program.  | See section <a href="#">Initializing the Adabas Communication Environment</a> .   |
| 7    | Create the sample JCS job control for installing Adabas.  | See section <a href="#">Prepare the Installation Sample JCS for Editing</a> .   |
| 8    | Customize and run job ADAIOOAL to link the Adabas options table for installation customization.   | See section <a href="#">Modify, Assemble, and Link the Adabas Options Table</a> .   |
| 9    | Customize and catalog the two procedures ADAV vLIB and ADAV vFIL before placing them back in the procedure library. The following specific items must be customized: <ul style="list-style-type: none"> <li>■ file IDs for the database and libraries;</li> <li>■ volumes for libraries and database files;</li> <li>■ space allocation for database files</li> </ul> | See section <a href="#">Catalog Procedures for Defining Libraries and the Database</a> .  |
| 10   | Customize and run ADAFRM to allocate and format the Adabas database.  | Steps 10-19 require changes to the setup definitions as described in section <a href="#">Database Installation Steps</a> .  |
| 11   | Customize and run ADADEF to define global database characteristics.   |   |
| 12   | Customize and run ADALODE, ADALODV, and ADALODM to load the demo files.   |   |
| 13   | Install the product license file.   |   |

| Step | Description  | Additional Information   |
|------|--|--|
| 14   | Customize and run ADANUC to start the Adabas nucleus to test Adabas communications.                      |  |
| 15   | Customize and run ADAREP in MULTI mode with the CPLIST parameter to test Adabas partition communication. |  |
| 16   | Customize and run ADAINPL to load the Adabas Online System, if used.                                     |  |
| 17   | Terminate the Adabas nucleus.  |  |
| 18   | Customize and run ADASAV to back up the database.  |  |
| 19   | Customize and run DEFAULTS to insert the ADARUN defaults with the ZAP utility.                           |  |
| 20   | Install the required TP link routines for Adabas   | See section <a href="#">Installing Adabas With TP Monitors</a> . |

## Contents of the Release Tape

The following table describes most of the libraries included on the release (installation) medium. Once you have unloaded the libraries from the medium, 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/VSE environments.

| Library Name       | Description  |
|--------------------|--|
| 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 <i>version</i> of Adabas.   |
| 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 <i>version</i> 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 <i>version</i> of Adabas.  |
| ADA $vrs$ .LC $nn$ | The Adabas library containing character encoding members to support various languages and Unicode. The $nn$ letters in the library name represents a number from "00" to "99", assigned by Software AG. The $vrs$ in the library name represents the <i>version</i> of Adabas.   |
| ADA $vrs$ .LIBR    | The source and sample job library for Adabas. The $vrs$ in the library name represent the <i>version</i> of Adabas. Sample jobs (*.X members) are stored in the SAGLIB.ADA $vrs$ sublibrary.<br><br>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 $vrs$ .MISC    | The Miscellaneous demo file, containing dummy miscellaneous data you can use for testing Adabas. The $vrs$ in the library name represents the <i>version</i> of Adabas.  |

| Library Name         | Description   |
|----------------------|---|
| ADA <i>vrs</i> .PERC | <p>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 <i>vrs</i> in the library name represents the <i>version</i> of Adabas.</p> <p><b>Note:</b> The Personnel demo file must be installed on a UES-enabled database because it includes wide-character format (W) fields.</p> |
| ADA <i>vrs</i> .TZ00 | <p>The time zone library for Adabas. The <i>vrs</i> in the library name represents the <i>version</i> of Adabas. Adabas bases its time zone library on the time zones defined in the public domain <a href="#">tz database</a>, also known as the <i>zoneinfo</i> or <i>Olson</i> database.</p> <p>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<i>vrs</i>.LIBR).</p>      |
| ADA <i>vrs</i> .VEHI | The Vehicles demo file, containing dummy vehicle data you can use for testing Adabas. The <i>vrs</i> in the library name represents the <i>version</i> of Adabas.   |
| APS <i>vrs</i> .L018 | A Software AG internal library. The <i>vrs</i> in the library name represents the <i>version</i> of the internal library code, which is not necessarily the same as the version of Adabas.  |
| APS <i>vrs</i> .LIBR | A Software AG internal library. The <i>vrs</i> in the library name represents the <i>version</i> of the internal library code, which is not necessarily the same as the version of Adabas.  |
| MLC <i>vrs</i> .LIBJ | The sample job library for Software AG's common mainframe license check software. The <i>vrs</i> in the library name represents the <i>version</i> of the license check software, which is not necessarily the same as the version of Adabas.   |
| MLC <i>vrs</i> .LIBR | The load library for Software AG's common mainframe license check software. The <i>vrs</i> in the library name represents the <i>version</i> of the license check software, which is not necessarily the same as the version of Adabas.   |
| WAL <i>vrs</i> .LIBR | The library for Adabas components shared by Adabas and other Software AG products, such as Entire Net-Work. The <i>vrs</i> in the library name represents the <i>version</i> of Adabas.   |
| WCA <i>vrs</i> .LIBR | The load library for Entire Net-Work Administration, used by some of the Adabas add-on products. The <i>vrs</i> in the library name represents the version of Entire Net-Work Administration.   |

## Preparing to Install Adabas

The major steps in preparing for Adabas installation are

- checking for the correct prerequisite system configuration; and
- allocating disk and storage space.

The following sections describe the nominal disk and storage space requirements, and how to allocate the space.

- [Disk Space Requirements for Libraries](#)



- [Disk Space Requirements for the Database](#)
- [Data Sets Required for UES Support](#)
- [Disk Space Requirements for Internal Product Data Sets](#)
- [Adabas Nucleus Partition/Address Space Requirements](#)
- [Defining the Library](#)
- [Restoring the ADAvrs LIBR File](#)
- [Using the ADAvrs LIBR File](#)

## Disk Space Requirements for Libraries

The Adabas library requires a minimum of 3390 disk space as shown below. A certain amount of extra free space has been added to the requirements for library maintenance purposes.

| Library        | 3390 Tracks |
|----------------|-------------|
| Adabas Library | 600         |

This space is needed for Adabas objects and phases as well as source and JCS samples.

## Disk Space Requirements for the Database

The Adabas database size is based on user requirements. For more information, refer to *Adabas DBA Tasks*. Suggested sizes for an initial Adabas database, allowing for limited loading of user files and the installation of Natural, are as follows.

The minimum 3390 disk space requirements are:

| Database Component            | 3390 Cylinders Required | 3390 Tracks Required |
|-------------------------------|-------------------------|----------------------|
| 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                  |

## Data Sets Required for UES Support

The Software AG internal product libraries (APS - porting platform) are required if you intend to enable a database for universal encoding service (UES) support. These libraries are now delivered separately from the product libraries.

For UES support, the following libraries must be loaded and included in the LIBDEF concatenation:

```
APSVrs.LIBR  
APSVrs.L0nn
```

where *vrs* is the *version* of the library provided on the most recent installation medium for these components and *aa* is LD, LC, or LS and *nn* is the load library level. If the library with a higher level number is not a full replacement for the lower level load library(s), the library with the higher level must precede those with lower numbers in the LIBDEF concatenation.

Also for UES support, the following library must be loaded and included in the session execution JCL:

```
ADAvrsCS.LIBR
```

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 Internal Product Data Sets

The minimum disk space requirements on a 3390 disk for the internal product libraries delivered with Adabas Version 8 are as follows:

| Library       | 3390 Cylinders | 3390 Tracks |
|---------------|----------------|-------------|
| ADAvrsCS.LIBR | 32             | 480         |
| APSVrs.LIBR   | 8              | 120         |
| APSVrs.L0nn   | 5              | 75          |

### Adabas Nucleus Partition/Address Space Requirements

The Adabas nucleus requires at least 900-1024 KB to operate. The size of the nucleus partition may need to be larger, depending on the ADARUN parameter settings. Parameter settings are determined by the user.

### Defining the Library

It is necessary to define the library before restoration. The following two examples show how VSAM and non-VSAM libraries are defined.

#### Defining a VSAM Library

The following is a job for defining a VSAM library:

```
// JOB DEFINE DEFINE VSAM V8 ADABAS LIBRARY
// OPTION LOG
// EXEC IDCAMS,SIZE=AUTO
DEFINE CLUSTER -
(NAME(ADABAS.ADAvrs.LIBRARY) -
VOLUME(vvvvvv vvvvvv) -
NONINDEXED -
RECORDFORMAT(NOCIFORMAT) -
SHR(2) -
TRK(nnnnnn)) -
DATA (NAME(ADABAS.ADAvrs.LIBRARY.DATA))
/*
// OPTION STDLABEL=ADD
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY',,VSAM
// EXEC IESVCLUP,SIZE=AUTO
ADABAS.ADAvrs.LIBRARY
/*
// EXEC LIBR
DEFINE L=SAGLIB R=Y
DEFINE S=SAGLIB.ADAvrs REUSE=AUTO R=Y
LD L=SAGLIB OUTPUT=STATUS
/*
/&

--where
vvvvvv vvvvvv are the volumes for primary and secondary space.
nnnnnn is the number of tracks for primary and secondary space.
vrs is the Adabas version.
```



#### Notes:

1. For FBA devices the tracks (TRK...) operand is replaced by the blocks (BLOCKS...) operand.
2. SAGLIB is the name of the Adabas library. The name SAGLIB can be changed to suit user requirements.

### Defining a Non-VSAM Library

The following is a job for defining a non-VSAM library:

```
// JOB DEFINE DEFINE NON-VSAM V8 ADABAS LIBRARY
// OPTION LOG
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY',2099/365,SD
// EXTENT SYS010,vvvvvv,1,0,ssss,nnnn
// ASSGN SYS010,DISK,VOL=vvvvvv,SHR
// EXEC LIBR
DEFINE L=SAGLIB R=Y
DEFINE S=SAGLIB.ADAvrs REUSE=AUTO R=Y
LD L=SAGLIB OUTPUT=STATUS
```

```
/*  
/&  
  
where:  
SYS010 is the logical unit for Adabas library.  
vvvvvv is the volume for Adabas library.  
ssss is the starting track or block for specified library.  
nnnn is the number of tracks or blocks for specified library.  
vrs is the Adabas version.
```

## Restoring the ADAvrs LIBR File

Restore the ADA<sub>vrs</sub> LIBR file into sublibrary SAGLIB.ADA<sub>vrs</sub>. See the next section for information about preparing modules to run without the ESA option active.



**Note:** See the *Software AG Product Delivery Report* that accompanies the installation media to position the media to the correct file.

If you have a license for one of the following Software AG products, restore the file into the appropriate sublibrary:

| Product                                   | File                     | Sublibrary                |
|---|--------------------------|---------------------------|
| Adabas Caching Facility (ACF)             | ACF <sub>vrs</sub> .LIBR | SAGLIB.ACF <sub>vrs</sub> |
| Adabas Online System (AOS)                | AOS <sub>vrs</sub> .LIBR | SAGLIB.AOS <sub>vrs</sub> |
| Adabas Parallel Services (ASM)            | ASM <sub>vrs</sub> .LIBR | SAGLIB.ASM <sub>vrs</sub> |
| Adabas Delta Save Facility Facility (ADE) | ADE <sub>vrs</sub> .LIBR | SAGLIB.ADE <sub>vrs</sub> |

For information about installing these products, see the documentation for that product.

## Using the ADAvrs LIBR File

Where applicable, modules for Adabas are shipped with AMODE=31 active.

### Storage Above or Below the 16-MB Limit

Adabas can acquire storage above the 16-megabyte addressing limit. This capability allows Adabas to acquire the buffer pool (LBP), work pool (LWP), format pool (LFP), and attached buffers (*NAB*) above 16 MB.

Where applicable, modules for Adabas are shipped with AMODE=31 active. If you prefer to have buffers placed below the 16-megabyte limit, ADARUN must be relinked with AMODE=24.

### User Program Execution in AMODE=31 and RMODE=ANY

Programs that will execute AMODE=31 or RMODE=ANY must be relinked with the new ADAUSER object module.

In addition, because the IBM VSE LOAD macro cannot be issued in `RMODE=ANY`, the IBM VSE CDLOAD macro must be used. Therefore, the zap to change the ADAUSER CDLOAD to the LOAD macro cannot be used.

## Initializing the Adabas Communication Environment

---

Communication between the Adabas nucleus residing in a z/VSE partition and the user (either a batch job or TP monitor such as Com-plete or CICS) in another partition is handled with an Adabas SVC (supervisor call).

The program ADASIP is used to install the Adabas SVC. The system can run ADASIP to dynamically install the SVC without an IPL. Special instructions apply when using z/VSE with the Turbo Dispatcher as described in the next section below.

For information about messages or codes that occur during the installation, refer to the *Adabas Messages and Codes* documentation.

- [Installing the Adabas SVC with Turbo Dispatcher Support](#)
- [ADASIP Processing](#)
- [Running ADASIP](#)
- [Finding an Unused SVC](#)
- [Loading a Secondary Adabas SVC](#)
- [ADASIP Execution Parameters](#)
- [ADASIP Runtime Display](#)

### Installing the Adabas SVC with Turbo Dispatcher Support

The Adabas SVC module supports the IBM z/VSE Turbo Dispatcher environment.

In a Turbo Dispatcher environment, the Adabas SVC runs in parallel mode when entered. Adabas processes multiple SVC calls made by users in parallel.

### ADASIP Processing

To enable Turbo support, ADASIP installs a z/VSE first-level interrupt handler (ADASTUB) that screens all SVCs. When ADASTUB finds an Adabas SVC, it passes control directly to the Adabas SVC.

If your system is capable of running the Turbo Dispatcher and you do not want to run a particular SVC through the Turbo interface, you can set the UPSI flag V to 1 to exclude a particular SVC from use through the Turbo interface. See the ADASIP UPSI statement.

You can activate the ADABAS SVC with multiple CPUs active by specifying UPSI C. ADASIP will dynamically de-activate and re-activate the CPUs if required. If multiple CPUs are active and the UPSI C has not been specified, the following messages will be displayed:

```
ADASIP60 Only 1 CPU can be active during ADASIP
ADASIP79 Should we stop the CPUs? (yes/no)
```

Answering yes to this message will allow activation to occur; the CPUs will be dynamically de-activated and re-activated. Answering no will terminate ADASIP.

The ADASTUB module is installed only once per IPL process. On the first run of a successful ADASIP, the following set of messages are returned:

```
ADASIP63 ADASTUB Module Loaded at nnnnnnnn
ADASIP78 VSE Turbo Dispatcher Version nn
ADASIP69 Turbo Dispatcher Stub A C T I V E
```

When running ADASIP for subsequent Adabas SVC installations, the following message is displayed for information only:

```
ADASIP74 Info : Stub activated by previous ADASIP
```

When dynamically re-installing an Adabas SVC that was previously installed with Turbo Dispatcher support, execute a SET SDL for the Adabas SVC only. Do not execute the SET SDL for ADANCHOR a second time.



**Note:** Repeated re-installations of an Adabas SVC without an IPL may result in a shortage of 24-bit GETVIS in the SVA.

## Running ADASIP

ADASIP requires a prior SET SDL for the SVC, and therefore must run in the BG partition. To install the Adabas SVC without an IPL, execute the following JCS in BG.



### Notes:

1. When using the EPAT Tape Management System, EPAT must be initialized before running ADASIP.
2. At execution time, the ADASIP program determines if a printer is assigned to system logical unit SYSLST. If no printer is assigned, messages are written to SYSLOG instead of SYSLST.

For information about the ADASIP parameters, see the section [ADASIP Execution](#).

To automatically install the Adabas SVC during each IPL, insert the following JCS (or its equivalent) into the ASI BG JCS procedure immediately before the START of the POWER partition where

|               |   |
|---------------|---|
| <i>nn</i>     | is the number of IDT entries  |
| <i>suffix</i> | is the optional two-byte suffix for the z/VSE SVC name to be loaded by ADASIP. The previous z/VSE SVC version must be linked with a different suffix. |
| <i>svc</i>    | is an available SVC number in your z/VSE system to be used as the Adabas SVC.   |
| <i>volume</i> | is the specified volume for the Adabas library.   |
| <i>vrs</i>    | is the Adabas <i>version</i>  |

### Without Turbo Dispatcher Support

The following sample is available in member ADASIP.X:

```
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY'
// EXTENT SYS010,volume
// ASSGN SYS010,DISK,VOL=volume,SHR
// LIBDEF PHASE,SEARCH=SAGLIB.ADAvrs
SET SDL
ADASVCvr,SVA
/*
// OPTION SYSPARM='svc,suffix' SVC NUMBER
// UPSI 00000000 UPSI OPTIONS FOR ADASIP
// EXEC ADASIP,PARM='NRIDTES=nn'
```

### With Turbo Dispatcher Support

The following sample is available in member ADASIPT.X:

```
// JOB ADASIPT INSTALL THE ADABAS SVC (TURBO)
// OPTION LOG,NOSYSDUMP
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY'
// EXTENT SYS010,volume
// ASSGN SYS010,DISK,VOL=volume,SHR
// LIBDEF PHASE,SEARCH=SAGLIB.ADAvrs
SET SDL
ADASVCvr,SVA
ADANCHOR,SVA
/*
// OPTION SYSPARM='svc,suffix' SVC NUMBER
// SETPFIX LIMIT=100K REQUIRED; SEE NOTE 2
// UPSI 00000000 UPSI OPTIONS FOR ADASIP
// EXEC ADASIP,PARM='NRIDTES=nn'
```



#### Notes:

1. A SETPFIX parameter is required with Turbo Dispatcher support to page fix ADASIP at certain points in its processing. A value of 100K should be adequate.

2. The SET SDL statement for ADANCHOR is required for Turbo Dispatcher support. This is in addition to the SET SDL statement for ADASVC<sub>vr</sub>.

## Finding an Unused SVC

Adabas requires an entry in the z/VSE SVC table. To find an unused SVC, use one of the following methods:

### Method 1

Set the S flag specified in the UPSI for ADASIP to create a list of used and unused SVCs in the z/VSE SVC table.

### Method 2

Obtain a listing of the supervisor being used.

Using the assembler cross-reference, locate the label SVCTAB; this is the beginning of the z/VSE SVC table. The table contains a four-byte entry for each SVC between 0 and 150 (depending on the z/VSE version).

Locate an entry between 31 and 150 having a value of ERR21. This value indicates an unused SVC table entry. Use the entry number as input to ADASIP.

## Loading a Secondary Adabas SVC

You can optionally specify a suffix to indicate the version of an SVC, as shown in the previous JCS examples. This allows you to run two different versions of the SVC. Before specifying a suffix, however, you must have previously linked the second version of the SVC. In addition, you must have performed a SET SDL operation on the new SVC's name (for example, ADASVC<sub>xx</sub>).

To optionally specify a different Adabas SVC using ADASIP, specify the SVC suffix (the last two bytes in the form, ADASVC<sub>xx</sub>), as follows, where <sub>xx</sub> is the two-byte suffix of the new SVC:

```
// OPTION SYSPARM='svc,xx'
```



## ADASIP Execution Parameters

This section describes the ADASIP execution parameters.

- [OPTION SYSPARM= Statement](#)
- [UPSI Statement](#)
- [NRIDTES PARM= Option](#)
- [REPLACE PARM= Option](#)
- [DMPDBID PARM= Option](#)

## Runtime Display

### OPTION SYSPARM= Statement

An optional correction (zap) can be applied to the Adabas ADASIP program to insert the default SVC so that no SYSPARM need be specified. See the section [Applying Zaps](#).

|        |  |
|--------|--|
| SVC    | The Adabas SVC number chosen must be unused by z/VSE or any other third party products (see the section <a href="#">Finding an Unused SVC</a> ). |
| SUFFIX | An optional two-byte value used to load a new version of the Adabas z/VSE SVC (see the section <a href="#">Loading a Secondary Adabas SVC</a> ). |

### UPSI Statement

```
// UPSI DSxTVCGx
```

Setting the UPSI byte is the user's responsibility. If the UPSI byte is not set, the SVC installation executes normally.

The UPSI byte is used to select the following options:

| Option | If option is set to 1  |
|--------|--|
| D      | ADASIP dumps the Adabas SVC and ID table using PDUMP. This option should be used only after the SVC is installed.  |
| S      | ADASIP dumps the z/VSE SVC table and indicates whether each SVC is used or unused. No SVC number is required when using this function of ADASIP.   |
| T      | ADASIP dumps the z/VSE SVC table and the z/VSE SVC mode table.   |
| V      | The SVC is excluded from use through the Turbo interface.  |
| C      | Override the messages that ask if you wish to stop the processors when more than one processor is active. If you choose to override, the processors will be automatically stopped during ADASIP execution and restarted upon ADASIP termination. |
| G      | ADASIP will display SYSTEM GETVIS allocation totals.   |

### NRIDTES PARM= Option

The size of the ID table default supports up to 10 Adabas targets. However, the ADASIP program will allow you to increase this number by using this new option of the PARM operand on the EXEC card. To increase the size of the ID table to *nn* entries, specify the following when executing ADASIP:

```
// EXEC ADASIP,PARM='NRIDTES=nn'
```

where *nn* is the number of databases to be supported. Refer to the section [Acquiring Storage for the ID Table](#) for information about calculating the correct value for *nn*.

### REPLACE PARM= Option

Specifying REPLACE=N or NO will cause warning messages ADASIP80 and ADASIP81 to appear if the SVC has been previously installed. Specifying REPLACE=Y or YES replaces the current SVC regardless of any active targets. The default value is REPLACE=NO. No abbreviation of the REPLACE keyword is supported.



**Caution:** Setting the REPLACE parameter to YES should be done carefully. Replacing an SVC while your targets are running can produce unpredictable results.

If both the NRIDTES and REPLACE keywords are specified, they must be separated by a comma. For example:

```
//EXEC ADASIP,PARM='NRIDTES=10,REPLACE=YES'
```

### DMPDBID PARM= Option

This ADASIP option allows snap dumps of the Adabas command queue for a specified database ID (DBID). The dump is written to SYSLST. The OPTION SYSPARM statement must specify the SVC number to perform the snap dump. For example, to perform a snap dump of the database 5 command queue, issue:

```
// OPTION SYSPARM='svc,suffix'  
// EXEC ADASIP,PARM='DMPDBID=5'
```

## ADASIP Runtime Display

When ADASIP is run, the ADASIP00 message displays the current system level.

```

ADASIP00 ...ADABAS V8 VSE SIP STARTED
SIP IS RUNNING UNDER VSE/systype-mode
ADASIP00 ... (yyyy-mm-dd, SM=sm-level, ZAP=zap-level)
ADASIP00 ... SIP IS RUNNING UNDER OSYS LEVEL Vnnn
ADASIP00 ... SIP IS LOADING ADABAS SVC LEVEL Vnnn
ADASIP00 ... ADASIP IS LOADING ADABAS SVC AMODE=amode

```

## Installing the Adabas Database

This section describes installation of the Adabas database for z/VSE systems. Note that all applicable early warnings and other fixes must first be applied. For descriptions of any messages or codes that occur, refer to the *Adabas Messages and Codes* documentation.

- [Installing the Release Tape](#)
- [Prepare the Installation Sample JCS for Editing](#)
- [Modify, Assemble, and Link the Adabas Options Table](#)
- [Catalog Procedures for Defining Libraries and the Database](#)
- [Database Installation Steps](#)

### Installing the 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.

This section explains how to copy the data sets .LIBJ, .LIBR and .LICS from tape to disk. All other data sets can be installed directly from the tape.

- [Step 1: Copy Data Set COPYTAPE.JOB to Disk](#)
- [Step 2: Modify COPYTAPE.JOB on Your Disk](#)

- [Step 3: Submit COPYTAPE.JOB](#)

**Step 1: Copy Data Set COPYTAPE.JOB to Disk**

- Modify the following sample job according to your requirements:

```
* $$ JOB JNM=LIBRCAT,CLASS=0,                                     +
* $$ DISP=D,LDEST=(*,UID),SYSID=1
* $$ LST CLASS=A,DISP=D
// JOB LIBRCAT
* *****
*      STORE COPYTAPE.JOB IN LIBRARY
* *****
// ASSGN SYS004,nnn
// MTC REW,SYS004
// MTC FSF,SYS004,4
ASSGN SYSIPT,SYS004
// TLBL IJSYSIN,'COPYTAPE.JOB'
// EXEC LIBR,PARM='MSHP; ACC S=lib.sublib'
/*
// MTC REW,SYS004
ASSGN SYSIPT,FEC
/*
/&
* $$ E0J
```

where:

*nnn* is the tape address, and

*lib.sublib* is the library and sublibrary in which the data set COPYTAPE.JOB is to be stored.

- Execute the job to copy the data set COPYTAPE.JOB to disk.

COPYTAPE.JOB contains the JCL required to copy the data sets .LIBJ, .LIBR and .LICS from tape to disk.

**Step 2: Modify COPYTAPE.JOB on Your Disk**

- Modify COPYTAPE.JOB according to your requirements and set the disk space parameters as appropriate.

### Step 3: Submit COPYTAPE.JOB

- Execute COPYTAPE.JOB to copy the data sets .LIBJ, .LIBR and .LICS to your disk.

### Prepare the Installation Sample JCS for Editing



**Note:** This step is only necessary if the library cannot be edited directly.

The following sample installation job is available in member INSTALL.X.

Run the following job to load the installation samples:

```
* $$ JOB JNM=PUNINST,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
* $$ PUN CLASS=p,DISP=D
// JOB PUNINST INSTALL SAMPLES FOR ADABAS
// OPTION LOG
// DLBL SAGLIB,'ADABAS.ADAvrs.LIBRARY'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volume,SHR
// EXEC LIBR
ACCESS SUBLIB=SAGLIB.ADAvrs
PUNCH ADAPROC.X /* PROCS FOR FILE AND LIBRARY DEFINITIONS */
PUNCH ADAIOOAL.X /* ADABAS OPTIONS TABLE CUSTOMIZATION */
PUNCH ADASIP.X /* ADASIP JOB (NON-TURBO DISPATCHER) */
PUNCH ADASIPT.X /* ADASIP JOB (TURBO DISPATCHER) */
PUNCH ADAFRM.X /* SAMPLE ADAFRM JOB */
PUNCH ADADEF.X /* SAMPLE ADADEF JOB */
PUNCH ADALODE.X /* LOAD DEMO FILE EMPLOYEES */
PUNCH ADALODV.X /* LOAD DEMO FILE VEHICLES */
PUNCH ADALODM.X /* LOAD DEMO FILE MISC */
PUNCH ADALODP.X /* LOAD DEMO FILES PERSONNEL & LOB */
PUNCH ADANUC.X /* SAMPLE NUCLEUS STARTUP */
PUNCH ADAREP.X /* SAMPLE ADAREP JOB */
PUNCH ADAINPL.X /* SAMPLE NATINPL TO INSTALL AOS */
/*
/&
* $$ EOJ
```

where *p* is the output class for punch, *volume* is the specified volume for the Adabas library, and *vrs* is the Adabas *version*.

Once the selected members in the INSTALL job are within the local editor facility, the customization can begin.

## Modify, Assemble, and Link the Adabas Options Table

Customize and run job ADAIOOAL to assemble and link the Adabas options table for installation customization.

The following describes the IORDOSO macro, which must be assembled and linked to the Adabas sublibrary as PHASE ADAOPD. The member X.ADAIOOAL shipped with Adabas can be used for this purpose.

- [IORDOSO Macro Overview](#)
- [IORDOSO Macro Parameters](#)

### IORDOSO Macro Overview

The IORDOSO macro allows you to customize Adabas operation in the following areas:

- Loading phases;
- IDRC compaction support for 3480 and 3490 tape devices;
- Interfaces to z/VSE disk space managers such as DYNAM/D;
- Interfaces to z/VSE tape managers such as DYNAM/T
- An option controlling how the system writes to fixed block addressing (FBA) devices;
- An option to write printer (PRINT and DRUCK) files under either DTFPR or DTFDI control;
- GETVIS message printing;
- Optional job exit processing;
- Options for controlling the creation of z/VSE JCS with the Adabas Recovery Aid utility ADARAI;
- Sequential file processing under VSAM/SAM;
- Input device control with SYS000 assignment;
- Name of external sort program.

### IORDOSO Macro Parameters

The following parameters can be set in using the IORDOSO macro.

**CDLOAD**

| Parameter                | Description   |
|--------------------------|---|
| CDLOAD={ NO   YES }<br>↵ | Determines whether Adabas uses the CDLOAD (SVC 65) or the LOAD SVC (SVC 4) to load modules. |

**COMPACT**

| Parameter              | Description  |
|------------------------|--|
| COMPACT={ NO   YES } ↵ | If a sequential protection log (SIBA) is assigned to a 3480 or 3490 tape device, COMPACT=YES writes the SIBA in IDRC compaction mode. The default is COMPACT=NO (no compaction). |

**DISKDEV**

| Parameter            | Description   |
|----------------------|---|
| DISKDEV=devtype<br>↵ | Specifies the device type on which space for sequential files is to be allocated (see notes 1 and 2). |

**Notes:**

1. Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.
2. Valid disk device types are 3390, 9345 and FBA.

**DISKMAN**

| Parameter                 | Description  |
|---------------------------|--|
| DISKMAN={ NO   YES }<br>↵ | Indicates to Adabas that a z/VSE disk space manager such as DYNAM/D is active. If DISKMAN=YES is specified, DISKDEV or DISKSYS must also be specified. |

## DISKSYS

| Parameter                   | Description  |
|-----------------------------|--|
| DISKSYS= <i>sysnum</i><br>↵ | When a disk space manager such as DYNAM/D is present, use the DISKSYS parameter to specify the programmer logical unit (LUB). The specified value, which can be from 000 to 255, determines the disk device type for the SAM or VSAM sequential file. There is no default value. |

## DISKTYP

| Parameter            | Description  |
|----------------------|--|
| DISKTYP= <i>text</i> | This parameter is for information only, and is processed as a comment. The value <i>text</i> can be up to 16 bytes long. |

## DTFDI

| Parameter                      | Description   |
|--------------------------------|---|
| DTFDI={ <u>NO</u>   YES }<br>↵ | DTFDI=YES directs the PRINT (SYSLST) and DRUCK (SYS009) output to be device-independent, causing all ADARUN, ADANUC, session, statistics, and utility output to be written to where SYSLST or SYS009 is assigned (printer, disk, or tape). When you specify DTFDI=YES, the PRTDSYS and PRTRSYS parameters are ignored. If you specify DTFDI=NO (the default), output is directed using DTFPR. |

## FBAVRF

| Parameter                       | Description   |
|---------------------------------|---|
| FBAVRF={ <u>NO</u>   YES }<br>↵ | FBA users only: the FBAVRF parameter specifies whether Adabas does WRITE VERIFY I/O commands, or normal WRITES. If FBAVRF=YES is specified, WRITE VERIFY I/Os are performed; the default is normal WRITE operation. |

## GETMSG

| Parameter                       | Description   |
|---------------------------------|---|
| GETMSG={ <u>NO</u>   YES }<br>↵ | Determines whether or not z/VSE ADAIOR GETMAIN (GETVIS) messages are printed. No printing is the default. |



**JBXEMSG**

| Parameter                         | Description  |
|-----------------------------------|--|
| JBXEMSG={ NO   <u>YES</u>   PRT } | The z/VSE parameter JBXEMSG determines whether job exit utility error messages are printed (JBXEMSG=PRT), displayed (JBXEMSG=YES, the default), or not presented (JBXEMSG=NO). |

**JBXIMSG**

| Parameter                         | Description  |
|-----------------------------------|--|
| JBXIMSG={ NO   YES   <u>PRT</u> } | The z/VSE parameter JBXIMSG determines whether job exit utility information messages are printed (JBXIMSG=PRT, the default), displayed (JBXIMSG=YES), or not presented (JBXIMSG=NO). |

**JOBEXIT**

| Parameter                   | Description   |
|-----------------------------|---|
| JOBEXIT={ <u>NO</u>   YES } | JOBEXIT=YES activates the Adabas job exit utility, allowing any *SAGUSER job control statements to override the normal job input. |

**PFIXRIR**

| Parameter                   | Description   |
|-----------------------------|---|
| PFIXRIR={ <u>NO</u>   YES } | Specifies whether or not ADAMPM is page fixed in storage during the nucleus initialization process. |

**PRTDSYS**

| Parameter  | Description   |
|--|---|
| PRTDSYS={ <i>sysnum</i> ↔<br>  <u>SYSLST</u> } ↔ | <p>Specifies the programmer logical unit (LUB), and may be any number 000 - 254. If specified, the <i>sysnum</i> value replaces the default where the ADARUN messages are printed, which is SYSLST.</p> <p>The value specified by <i>sysnum</i> must be assigned in the partition before running the ADARUN program. For example:</p> |

| Parameter | Description                                      |
|-----------|--|
|           | <pre>PRTDSYS=050 . // ASSGN SYS050,PRINTER</pre> |

**PRTSYS**

| Parameter                                  | Description  |
|--|--|
| PRTSYS={ <i>sysnum</i>   <u>SYS009</u> } ↔ | Specifies the programmer logical unit (LUB). If specified, this <i>sysnum</i> value replaces the default where the Adabas utility (DRUCK) messages are printed, which is SYS009. |

**RAIDASG**

| Parameter                     | Description  |
|-------------------------------|--|
| RAIDASG={ NO   <u>YES</u> } ↔ | RAIDASG=YES specifies that the Adabas Recovery Aid (ADARAI) is to create z/VSE disk ASSGN statements. Such statements are sometimes not needed with a z/VSE disk manager facility. |

**RAITASG**

| Parameter                     | Description  |
|-------------------------------|--|
| RAITASG={ NO   <u>YES</u> } ↔ | RAITASG=YES specifies that the Adabas Recovery Aid (ADARAI) is to create z/VSE tape ASSGN statements. Such statements are sometimes not needed with a z/VSE tape manager facility. |

**SORTPGM**

| Parameter                                  | Description   |
|--|---|
| SORTPGM={ <i>sortpgm</i>   <u>SORT</u> } ↔ | Specifies the name of the external sort program to be invoked during execution of the Adabas changed-data capture utility ADACDC. The default name is SORT. |

**SYS0000**

| Parameter                        | Description  |                             |  |      |       |      |       |      |       |
|----------------------------------|--|-----------------------------|--|------|-------|------|-------|------|-------|
| SYS0000={ <u>NO</u>   YES }<br>↩ | <p>If SYS0000=NO (the default) is specified, the ADARUN statements are read normally. If SYS0000=YES is specified, Adabas determines the correct DTF for opening, depending on where SYS000 is assigned, as follows:</p> <table> <tr> <th colspan="2">Medium - SYS000    DTF Type</th></tr> <tr> <td>Card</td><td>DTFCD</td></tr> <tr> <td>Disk</td><td>DTFSD</td></tr> <tr> <td>Tape</td><td>DTFMT</td></tr> </table> | Medium - SYS000    DTF Type |  | Card | DTFCD | Disk | DTFSD | Tape | DTFMT |
| Medium - SYS000    DTF Type      |  |                             |  |      |       |      |       |      |       |
| Card                             | DTFCD  |                             |  |      |       |      |       |      |       |
| Disk                             | DTFSD  |                             |  |      |       |      |       |      |       |
| Tape                             | DTFMT  |                             |  |      |       |      |       |      |       |

**TAPEDEV**

| Parameter                    | Description   |
|------------------------------|---|
| TAPEDEV= <i>devtype</i><br>↩ | Specifies the tape device type on which sequential files are written (see notes 1 and 2). |

**Notes:**

1. Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.
2. Valid tape device types are 2400, 3410, 3420, 3480 and 8809. For device types 3480, 3490, 3490E or 3590, specify TAPEDEV=3480.

**TAPEMAN**

| Parameter                        | Description  |
|----------------------------------|--|
| TAPEMAN={ <u>NO</u>   YES }<br>↩ | Indicates that a z/VSE tape manager such as DYNAM/T is active. If TAPEMAN=YES is specified, TAPEDEV or TAPESYS must also be specified. |

## TAPESYS

| Parameter              | Description  |
|------------------------|--|
| TAPESYS= <i>sysnum</i> | When a tape manager such as DYNAM/T is present, this parameter is used to specify the programmer logical unit (LUB). The specified value, which can be any value from 000 to 255, determines the tape device type for the sequential file (see note). There is no default value. |



**Note:** Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.

## TAPETYP

| Parameter            | Description  |
|----------------------|--|
| TAPETYP= <i>text</i> | This parameter is for information only, and is processed as a comment. The value <i>text</i> can be up to 16 bytes long. |

## VSAMDEV

| Parameter               | Description  |
|-------------------------|--|
| VSAMDEV= <i>devtype</i> | Specifies the disk device type on which VSAM/SAM space is to be allocated (see notes 1 and 2). |



### Notes:

1. Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.
2. Valid disk device types are 3390, 9345 and FBA.

## VSAMSEQ

| Parameter            | Description   |
|----------------------|---|
| VSAMSEQ={ NO   YES } | Specifies whether sequential files are to be under the control of VSAM/SAM software. If VSAMSEQ=YES is specified, either VSAMDEV or VSAMSYS must also be specified. |

## VSAMSYS

| Parameter              | Description  |
|------------------------|--|
| VSAMSYS= <i>sysnum</i> | Specifies the programmer logical unit (LUB). The specified value, which can range from 000 to 255, determines the device type for the sequential file written to VSAM/SAM space (see note). There is no default value. |



**Note:** Adabas requires device type information when opening files. However, there may be situations where the device cannot be determined before the open without additional operations; for example, when a z/VSE Disk Space Manager or Tape Manager is active, or when using VSAM/SAM sequential files. Adabas also determines the block size to be used for sequential I/O areas by device type.

### Additional Parameters Used for Internal Control Only

Three additional parameters are also available but are used only for internal control and should not be changed from their default settings unless otherwise specified by your Software AG technical support representative:

```
IORTRAC={NO | YES}
IORTSIZ={3000 | tablesize}
IORTTYP=(1... 14)(, opt1 ... opt14 ).
```

### Catalog Procedures for Defining Libraries and the Database



**Note:** Sample JCS is available in ADAPROC.X

The job ADAPROC is divided into two procedures:

- ADAV vLIB defining the library or libraries; and
- ADAV vFIL defining the database.

Customize and catalog the two procedures before placing them back in the procedure library. The following specific items must be customized:

- file IDs for the database and libraries;

- volumes for libraries and database files;
- space allocation for database files.

The Adabas DEMO database files include ASSO, DATA, WORK, TEMP, SORT, CLOG, and PLOG.

### Database Installation Steps

Follow the steps outlined below to install a new Adabas database under z/VSE.

- Step 1. Allocate and format the DEMO database.
- Step 2. Define the global database characteristics.
- Step 3. Load the demonstration (demo) files.
- Step 4. Install the product license file.
- Step 5. Start the Adabas nucleus and test the Adabas communications.
- Step 6. Test Adabas partition communications.
- Step 7. Load the Adabas Online System, if used.
- Step 8. Terminate the Adabas nucleus.
- Step 9. Back up the database.
- Step 10. Insert the ADARUN defaults.
- Step 11. Install the required TP link routines for Adabas.



#### Notes:

1. For information about running ADADEF, ADAFRM ADALOD, ADAREP, and ADASAV in steps 1-3, 5, and 8 below, see the *Adabas Utilities* documentation.
2. For information about customizing the nucleus job and about starting, monitoring, controlling, and terminating the nucleus, see the *Adabas Operations* documentation.

#### Step 1. Allocate and format the DEMO database.



**Note:** Sample JCS is available in ADAFRM.X

Customize and run the ADAFRM utility job to format the DEMO database areas. The following specific items must be customized:

- the Adabas SVC number, the database ID, and database device type(s);
- sizes of the data sets for each ADAFRM statement.

**Step 2. Define the global database characteristics.**

**Note:** Sample JCS is available in ADADEF.X

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

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

**Step 3. Load the demonstration (demo) files.**

**Note:** Sample JCS is available in ADALODE.X, ADALODV.X, ADALODM.X, and ADALODP.X.

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:

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

**Step 4. Install the product license file.**

The product license file is supplied on the individual customer installation media or separately via an e-mail attachment. If the license file is provided with the installation media, you can follow the instructions in this step to install the license file. If the license file is supplied via an e-mail attachment, you must first transfer the license to z/VSE, as described in *Transferring a License File from PC to a z/VSE Host Using FTP*, in *Software AG Mainframe Product Licensing* and then you can install it, as described in this step.

**Installing the license file.**

In z/VSE environments, the product license file can be installed either as a load module or as a library member.

➤ To install the product license file as a module, complete the following steps:

- 1 Verify that the license file is stored in an Adabas source library or sequential data set (with RECFM=F or FB and LRECL=80), taking care to preserve its format as ASCII.
- 2 If you loaded your Adabas license file to a library, review and modify sample JCS job ASMLICAL.X, adjusting the library and volume specifications as appropriate for your site. If you loaded your Adabas license file to a data set, use sample ASMLICAV.X instead.



**Note:** In sample jobs ASMLICAL.X and ASMLICAV.X, the standard label area is assumed to contain label information for library USERLIB. You can change this as appropriate for your library.

- 3 Submit modified sample job ASMLICAL.X or ASMLICAV.X.

These sample jobs generate your Adabas license in ADALIC.PHASE. They assume that ADALIC.PHASE will be in a user sublibrary. If a user sublibrary is chosen for ADALIC.PHASE, this sublibrary must be included in the LIBDEF search chain in your Adabas nucleus startup JCS. You may find it more convenient to place ADALIC.PHASE directly into the Adabas ADAvrs sublibrary, to avoid the need to define additional libraries. During initial testing, Software AG recommends using a user sublibrary.

➤ To install the product license file as a library member, complete the following steps:

- 1 Verify that the 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.
- 2 Make sure any previously created ADALIC load module is inaccessible in the Adabas load library being used by the nucleus jobs. Adabas first tries to load ADALIC and if unsuccessful it reads from DDLIC.
- 3 Provide all Adabas nucleus startup jobs with a DLBL statement in the following format:

```
// DLBL DDLIC,'//libname/sublib/memname.memtype'
```

where *libname* is the Librarian name of the library, *sublib* is the name of the sublibrary, *memname* is the license member name, and *memtype* is the license member type.

➤ To install the product license file as a sequential data set, complete the following steps:

- 1 Verify that the license file is stored in a sequential file (with RECFM=F or FB and LRECL=80), taking care to preserve its format as ASCII.
- 2 Make sure any previously created ADALIC load module is inaccessible in the Adabas load library being used by the nucleus jobs. Adabas first tries to load ADALIC and, if unsuccessful, it reads from DDLIC.



- 3 Provide all Adabas nucleus startup jobs with DLBL, EXTENT and ASSGN statements in the following format:

```
// DLBL DDLIC,'adabas.license.file'
// EXTENT SYSnnn
// ASSGN SYSnnn,DISK,VOL=volser,SHR
```

where *adabas.license.file* is the physical file name, *nnn* is an unused logical unit, and *volser* is the volume serial on which the license file resides.

#### Step 5. Start the Adabas nucleus and test the Adabas communications.



**Note:** Sample JCS is available in ADANUC.X.

Customize and run the job ADANUC to start up the Adabas nucleus. The following items must be customized:

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



**Note:** Be sure to include appropriate LIBDEF references for user sublibraries, especially the library containing the ADALIC license file. The licensing component *MLC<sub>vrs</sub>* must also be added to the LIBDEF SEARCH chain for load modules. These additional sublibraries can be added via the ADAV<sub>v</sub>FIL procedure, as required.

- ADANUC parameters.

#### Step 6. Test Adabas partition communications.



**Note:** Sample JCS is available in ADAREP.X.

Customize and run the job ADAREP in MULTI mode with the *CPLIST* parameter to test Adabas partition communications. The following items must be customized:

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

### Step 7. Load the Adabas Online System, if used.



**Note:** Sample JCS is available in ADAINPL.X. Read [Installing the AOS Demo Version](#), elsewhere in this guide, and, if necessary, the installation section of the Adabas Online System documentation.

Customize and run the job ADAINPL to load the Adabas Online System into a Natural system file. A Natural file must first be created, requiring an INPL input file (see the Natural installation instructions). The following items must be customized:

- the Adabas SVC number, the database ID, and device type(s);
- the Natural INPL parameters and system file number.

### Step 8. Terminate the Adabas nucleus.

Communicate with the Adabas nucleus (MSG F*n*) to terminate the session by entering the Adabas operator command ADAEND into the Adabas nucleus partition.

### Step 9. Back up the database.

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

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

### Step 10. Insert the ADARUN defaults.

Optionally customize and run the DEFAULTS job to set the ADARUN defaults using the MSHP utility and to relink ADARUN. The following items may be customized:

- SVC number;
- database ID;
- device type(s).

**Step 11. Install the required TP link routines for Adabas.**

Refer to the section [Installing Adabas With TP Monitors](#) for the TP link routine procedure.

## Migrating an Existing Database

---

Use the ADACNV utility to migrate existing databases to new releases of Adabas. See the *Adabas Utilities* documentation for more information.

## Logical Unit Requirements

---

This section describes the Adabas logical unit requirements.

**ADARUN**

| Logical Unit | File  | Storage Medium |
|--------------|-------|----------------|
| SYSLST       | PRINT | Printer        |
| SYS000       | CARD  | Tape / Disk    |
| SYSRDR       | CARD  | Reader         |

**Utility**

| Logical Unit | File  | Storage Medium |
|--------------|-------|----------------|
| SYS009       | DRUCK | Printer        |
| SYSIPT       | KARTE | Reader         |

**Nucleus**

| Logical Unit | File  | Storage Medium |
|--------------|-------|----------------|
| SYSLST       | PRINT | Printer        |
| SYSRDR       | CARD  | Reader         |

The highest logical unit used is SYS038 for the ADASAV utility. The programmer logical units default is described in the section [Device and File Considerations](#). The system programmer should review these requirements to ensure that there are enough programmer logical units to run the desired utilities in the desired partitions.

## Job Exit Utility

---

Adabas provides a job exit to perform two different functions:

- Librarian input override processing

The exit scans a job stream for Librarian input override statements. These statements indicate that card input (ADARUN CARD or utility KARTE statements) for a job step is to come from Librarian members rather than from SYSRDR or SYSIPT.

- ADARAI JCS capture processing

The exit captures JCS before it is modified by tape or disk management systems for later use by ADARAI.

You can set the job exit to perform either function or both. By default, the job exit performs Librarian input override processing.

This section covers the following topics:

- [Installation and Initialization](#)
- [Librarian Input Override Processing](#)
- [Activating Adabas Use of Job Exit Processing](#)
- [Using the Job Exit Utility for ADARAI JCS Capture](#)
- [Job Exit Storage Requirements](#)
- [Optional Console or Printer Messages](#)
- [Diagnostic Functions](#)

### Installation and Initialization

The job exit can be installed during ASI processing or at any time afterward. It is installed in two steps:

➤ **to install the job exit:**

- 1 Install programs SAGJBXT and SAGIPT in the SVA.
- 2 Run program SAGINST to initiate job exit processing.

You can include SAGJBXT in the \$JOBEXIT list of eligible exits, but you must still place SAGIPT in the SVA and run SAGINST to allocate the required table(s).

SAGIPT runs above the 16-megabyte line if an appropriate 31-bit PSIZE is available. In addition, the table that stores information from input-override statements and/or the table that stores JCS for ADARAI use is placed in 31-bit GETVIS, if available.

SAGINST reads an input parameter that tells it whether to install the Librarian input override processing, ADARAI JCS capture processing, or both. The following parameter values are valid:

PARM=ADALIB (the default) installs Librarian input override processing  
 PARM=ADARAI installs ADARAI JCS capture processing

The following sample job control initializes the job exit:



**Note:** Sample JCS to initialize the job exit is available in member JBXTINST.X.

```
* $$ JOB JNM=SAGEXIT,CLASS=0
* $$ LST CLASS=A,DISP=D
// JOB SAGEXIT
// LIBDEF *,SEARCH=SAGLIB.ADAvrs
// EXEC PROC=ADAVvLIB
SET SDL
SAGJBXT,SVA
SAGIPT,SVA
/*
// EXEC SAGINST,PARM=ADARAI,ADALIB
/&
* $$ EOJ
```

where *vrs* is the Adabas *version*.

## Librarian Input Override Processing

If Librarian input override processing is specified, the job exit scans a job stream for input override statements indicating that card input (ADARUN CARD or utility KARTE statements) for a job step is to come from Librarian members rather than from SYSRDR or SYSIPT. By default, the exit can store a maximum of 2000 input override cards simultaneously throughout the system. Adabas uses this facility when processing CARD and KARTE parameters.

Enable Librarian input override processing by adding \* SAGUSER control statements to the job control stream between the // JOB and // EXEC statements.

A \* SAGUSER statement can have three keyword parameters: FILE, LIBRARY, and MEMBER.

| Keyword Syntax  | Description   |
|---|---|
| FILE={ CARD   KARTE }   | The file to be read from a Librarian member. Specify "CARD" for ADARUN statements, or "KARTE" for utility statements. |
| LIBRARY={ <i>library.sublibrary</i> ↵<br>  <i>libdef.source</i> } | The library and sublibrary to be searched. If omitted, the current <i>libdef.source</i> chain is used.                |

| Keyword Syntax                               | Description  |
|--|--|
| MEMBER= <i>name</i> [, { <i>type</i>   A } ] | The member name and optionally the type to be read. If <i>type</i> is omitted, "A" is assumed. |

The following is an example of a \*SAGUSER control statement that specifies an alternate job exit member:

```
* SAGUSER FILE=CARD, MEMBER=NUC151
```

In the example above, Adabas searches the current *libdef.source* chain for member NUC151 with type A. If NUC151 is found, Adabas uses its contents as the nucleus startup parameters instead of SYSIPT.

To permit flexible startup processing, multiple SAGUSER statements may be specified for each file. In the following example, Adabas reads the input parameters first in member NUC151, then in member IGNDIB:

```
* SAGUSER FILE=CARD, MEMBER=NUC151
* SAGUSER FILE=CARD, MEMBER=IGNDIB
```

The following examples show the use of the LIBRARY parameter, and apply to z/VSE systems only:

```
* SAGUSER FILE=CARD, MEMBER=NUC151, LIBRARY=SAGULIB.TESTSRC
```

In the example above, Adabas searches sublibrary TESTSRC in the SAGULIB library for member NUC151 with type A. If NUC151 is not found in sublibrary TESTSRC of library SAGULIB, no further search is made. The DLBL and EXTENT information for the SAGULIB library must be available.

```
* SAGUSER FILE=CARD, MEMBER=NUC151.ADARUN, LIBRARY=SAGULIB.TESTSRC
```

In the example above, Adabas searches sublibrary TESTSRC in the SAGULIB library at nucleus initialization for member NUC151 with type ADARUN. The library member types PROC, OBJ, PHASE, and DUMP are not permitted.

### Activating Adabas Use of Job Exit Processing

Specify JOBEXIT=YES to allow Adabas to use SAGUSER statements in the job stream and recatalog the Adabas options table (ADAOPD).

## Using the Job Exit Utility for ADARAI JCS Capture

Once the job exit utility has been installed for ADARAI, all utilities that write information to the RLOG automatically obtain file information from the ADARAI table that the job exit maintains. Manual intervention is not required.

## Job Exit Storage Requirements

The job exit requires from 84 to 298 kilobytes (KB) of SVA storage, depending on whether the Librarian input override interface and/or the ADARAI JCS interface is installed. Of that total,

- 2 kilobytes are used for program storage (PSIZE);
- 82-kilobyte GETVIS for Librarian input override storage; and
- 214-kilobyte GETVIS for ADARAI JCS storage.

When running in z/VSE on z/VSE hardware, all of the GETVIS and 1 kilobyte of the PSIZE can be run above the 16-megabyte line.

## Optional Console or Printer Messages

You have the option of displaying, printing, or preventing these messages by specifying the JBXEMSG and JBXIMSG parameters in the Adabas options table.

## Diagnostic Functions

After the job exit is installed, you can produce dumps of the two tables for diagnostic purposes. Executing SAGINST with the ADASIP UPSI statement:

- UPSI 10000000 produces a dump of the Librarian input override table;
- UPSI 01000000 produces a dump of the ADARAI JCS table.

If the size of these two tables needs to be changed for any reason, SAGIPT may be zapped before being loaded into the SDL:

- The Librarian input override table size may be changed from the default of X'00014874' (84,084 bytes) to an appropriate value by zapping location X'18'. When altering the SAGIPT.OBJ module, ESDID=002 is required on the MSHP AFFECTS statement.
- The ADARAI JCS table size may be changed from the default of X'000355D6' to an appropriate value by zapping location X'0C'.

Each element in the Librarian input override table is 42 bytes in length. The default table size assumes 10 SAGUSER statements per file name, 10 file names, and 20 partitions, plus two extra unused entries. This number is an estimate of maximum concurrent residency; each statement is removed from the table after it is used.

Each element in the ADARAI JCS table is 91 bytes in length. The default table size accommodates 2400 entries with each DLBL, TLBL, or EXTENT statement requiring an entry in the table. Whenever a JOB statement is encountered, all entries for that partition (task ID) are cleared from the table.

## Acquiring Storage for the ID Table

---

The SYSTEM GETVIS is used to acquire storage for the ID table (IDT). This storage is acquired using the ADASIP at SVC installation time. The size of storage in the SYSTEM GETVIS depends on the number of IDT entries specified using ADASIP. The default number of IDT entries (IDTEs) is 10. The size can be calculated as follows:

```
SIZE (in bytes) = 1024 (IDT prefix) + 96 (IDT header) + (32 x number of IDTEs)
= 1024 + 96 + (32 x 10)
= 1024 + 96 + 320
= 1440 bytes
```

Also, additional SYSTEM GETVIS storage is acquired. This storage permits users to communicate from multiple address spaces when Adabas is not running in a shared partition. In this case, the following formula is used to calculate SYSTEM GETVIS:

```
SIZE (in bytes) = 192 (CQ header) + (192 x NC value) + (4352 x NAB value)
```

It may be necessary to increase the SVA size to meet these requirements. To do so, change the SVA operand in the appropriate \$IPL<sub>xxx</sub> procedure, then re-IPL.



**Note:** By default, the SYSTEM GETVIS is acquired above the 16-megabyte line. To acquire most of this space below the line, link-edit ADARUN AMODE 24.

## Acquiring Storage for the IIBS Table

---

The 31-bit SYSTEM GETVIS is used to acquire storage for the IIBS table (IIBS). This storage is acquired using the ADASIP at SVC installation time. The size of storage in the 31-bit SYSTEM GETVIS is 128K.



---

## SVC Work Areas

---

For each Adabas SVC installed, a number of 384-byte work areas are reserved. The number of work areas reserved is calculated as four times the number of IDTEs ( $4 \times IDTE-count$ ). The maximum number of work areas allocated is 128; the minimum is 24. The SVC work areas therefore occupy between 9K and 48K of storage. The default value of 10 IDTEs results in 15K of SYSTEM GETVIS being allocated.

---

## Displaying Storage Allocation Totals

---

Specifying `// UPSI xxxxxxGx` during the ADASIP execution (see UPSI byte description in [ADASIP Execution Parameters](#), earlier in this guide,) will generate allocation messages on the system console, showing the total 24-bit GETVIS and 31-bit GETVIS storage allocated by Adabas:

```
ADASIP85 GETVIS-24 storage allocated: nnnK
ADASIP85 GETVIS-31 storage allocated: nnnK
```

---

## Calls from Other Partitions

---

In order for an Adabas nucleus to accept calls from other partitions, storage is acquired in the SVA GETVIS area for any required attached buffers. The buffers hold data moved between the nucleus and users in other partitions.

---

## Dummy Sequential Files

---

If the file is not needed, it can be unassigned or assigned IGN such as the following:

```
// ASSGN SYS014,UA
```

or

```
// ASSGN SYS014,IGN
```

## Backward Processing of Tapes and Cartridges

---

To perform backward processing of tapes or cartridges, file positioning must occur before the file is opened. This can only be done when an assignment is made for the file. When performing the ADARES BACKOUT utility function, the // ASSGN ... for file BACK must be done explicitly.

No tape management system can be used, because such systems perform the assign operation when the file is opened; the LUB and PUB remain unassigned until this occurs.

## Applying Zaps (Fixes)

---

The jobs described in this section can be used to permanently change defaults and apply corrections (zaps) to the libraries in the supported z/VSE systems.

Two methods are used in z/VSE for applying corrective fixes to Adabas:

- the MSHP PATCH facility requires no definition of Adabas as a product/component on the MSHP history file. This method only alters phases. If the phase is relinked, the zap is lost.
- the MSHP CORRECT facility requires the definition of Adabas as a product/component using MSHP ARCHIVE.

Software AG distributes Adabas zaps to z/VSE users in MSHP CORRECT format and therefore recommends that you use MSHP CORRECT.

- [Applying Fixes Using MSHP PATCH](#)
- [Applying Fixes Using MSHP CORRECT](#)
- [Link Book Update Requirements for Secondary SVC](#)
- [Link Book Update Requirements for Running AMODE 24](#)

### Applying Fixes Using MSHP PATCH

A sample job for applying a fix to Adabas using MSHP PATCH is as follows:



**Note:** This sample job is available in member MSHPAT.X.

```
// JOB PATCH APPLY PATCH TO ADABAS
// OPTION LOG
// EXEC PROC=ADAVvLIB
// EXEC MSHP
PATCH SUBLIB=saglib.ADAvrs
AFFECTS PHASE=phasenam
ALTER offset vvvv : rrrr
/*
/ &
```

where

*vrs* is the Adabas version

*saglib* is the Adabas library name in procedure ADAVvFIL

*phasenam* is the Adabas phase to be zapped

*offset* is the hexadecimal offset into the phase

*vvvv* is the verify data for the zap

*rrrr* is the replace data for the zap

## Applying Fixes Using MSHP CORRECT

- [MSHP ARCHIVE](#)
- [MSHP CORRECT](#)

### MSHP ARCHIVE

For new users or users with no requirement to maintain multiple versions of Adabas, the following sample job can be used to define Adabas to MSHP.



**Note:** This job uses the history file identified by the IJSYSHF label in the z/VSE standard label area.



**Note:** This sample JCL is available in member MSHPARC.X.

```
// JOB ARCHIVE ARCHIVE ADABAS
// OPTION LOG
// EXEC PROC=ADAVvLIB
// EXEC MSHP
ARCHIVE ADAvrs
COMPRISES 9001-ADA-00
RESOLVES 'SOFTWARE AG - ADABAS Vv.r'
ARCHIVE 9001-ADA-00-vrs
RESIDENCE PRODUCT=ADAvrs -
PRODUCTION=saglib.ADAvrs -
GENERATION=saglib.ADAvrs
/*
/ &
```

```
—where  
vrs is the Adabas version  
saglib is the Adabas library name in procedure ADAVvFIL
```

A different MSHP history file must be used for each version and revision level of Adabas to which maintenance is applied.

To preserve the MSHP environment of an older version level of Adabas during an upgrade to a new version, it is necessary to create an additional MSHP history file for use by the new version.

The following sample MSHP job can be used to create an additional history file for a new version of Adabas and define Adabas to it.



**Note:** This sample JCL is available in member MSHPDEF.X.

```
// JOB ARCHIVE DEFINE HISTORY AND ARCHIVE ADABAS  
// OPTION LOG  
// EXEC PROC=ADAVvLIB  
// ASSGN SYS020,DISK,VOL=volhis,SHR  
// EXEC MSHP  
CREATE HISTORY SYSTEM  
DEFINE HISTORY SYSTEM EXTENT=start:numtrks -  
UNIT=SYS020 -  
ID='adabas.new.version.history.file'  
ARCHIVE ADAvrs  
COMPRISES 9001-ADA-00  
RESOLVES 'SOFTWARE AG - ADABAS Vv.r'  
ARCHIVE 9001-ADA-00-vrs  
RESIDENCE PRODUCT=ADAvrs -  
PRODUCTION=saglib.ADAvrs -  
GENERATION=saglib.ADAvrs  
/*  
/&  
  
—where  
vrs is the Adabas version  
volhis is the volume on which the Adabas Vvr history file resides  
start is the start of the extent on which the Adabas Vvr history file resides  
numtrks is the length of the extent on which the Adabas Vvr history file resides  
adabas.new.version.history.file is the physical name of the Adabas Vvr history file  
saglib is the Adabas library name in procedure ADAVvFIL
```

Once migration to the new version is complete, you can either

- continue to use the new history file to apply subsequent fixes; or
- delete the old version of Adabas from MSHP and merge the new version into the standard MSHP history file.



**Caution:** Before running any MSHP REMOVE or MERGE jobs, back up your MSHP environment by running MSHP BACKUP HISTORY jobs against all MSHP history files.

A sample MSHP job to remove an old version of Adabas is provided below.



**Note:** This sample JCL is available in member MSHPREM.X.

```
// JOB REMOVE REMOVE OLD ADABAS
// OPTION LOG
// PAUSE ENSURE MSHP HISTORY FILE BACKUP HAS BEEN TAKEN
// EXEC MSHP
REMOVE ADAvrs
REMOVE 9001-ADA-00-vrs
/*
/ &
```

—where *vrs* is the old Adabas version

A sample MSHP job to merge an additional history file for Adabas into the standard MSHP history file is provided below.



**Note:** This sample JCL is available in member MSHPMER.X.

```
// JOB MERGE MERGE SEPARATE ADABAS INTO STANDARD HISTORY
// OPTION LOG
// PAUSE ENSURE MSHP HISTORY FILE BACKUPS HAVE BEEN TAKEN
// ASSGN SYS020,DISK,VOL=volhis,SHR
// EXEC MSHP
MERGE HISTORY AUX SYSTEM
DEFINE HISTORY AUX EXTENT=start:numtrks -
UNIT=SYS020 -
ID='adabas.new.version.history.file'
/*
/ &
```

—where

*volhis* is the volume on which the Adabas Vvr history file resides

*start* is the start of the extent on which the Adabas Vvr history file resides

*numtrks* is the length of the extent on which the Adabas Vvr history file resides

*adabas.new.version.history.file* is the physical name of the Adabas Vvr history file

## MSHP CORRECT

The MSHP CORRECT and UNDO jobs use the history file identified by label IJSYSHF in the z/VSE standard label area. If Adabas is maintained from a different MSHP history file, include the following label information in the CORRECT or UNDO job:

```
// DLBL IJSYSHF,'adabas.new.version.history.file'  
// EXTENT SYSnnn  
// ASSGN SYSnnn,DISK,VOL=volhis,SHR  
  
-where  
volhis is the volume on which the Adabas Vvr history file resides  
nnn is the user-defined SYS number  
adabas.new.version.history.file is the physical name of the Adabas Vvr history file
```

A sample of the use of MSHP CORRECT to install a fix to Adabas is provided below.



**Note:** This sample JCL is available in member MSHPCOR.X.

```
// JOB CORRECT APPLY ADABAS FIX  
// OPTION LOG  
// EXEC PROC=ADAVLIB  
// EXEC MSHP  
CORRECT 9001-ADA-00-vrs : Axnnnnnn  
AFFECTS MODE=modname  
ALTER offset vvvv : rrrr  
INVOLVES LINK=lnkname  
/*  
/&  
  
-where  
vrs is the Adabas version  
x is the Adabas component (for example, N for nucleus)  
nnnnn is the Adabas fix number  
modname is the Adabas object module to be zapped and then relinked  
offset is the hexadecimal offset to the beginning of the zap  
vvvv is the verify data for the zap  
rrrr is the replace data for the zap  
lnkname is the link book for the phase affected
```

The CORRECT job updates object and phase in a single job step using the link book feature of MSHP. The INVOLVES LINK= statement automatically invokes the linkage editor after the object module is updated.

For a zap applied with the INVOLVES LINK= statement, the following UNDO can be used to remove the fix from both object module and phase:



**Note:** This sample JCL is available in member MSHPUND.X.

```
// EXEC MSHP
UNDO 9001-ADA-00-vrs : Axnnnnn
/*
```

where *vrs* is the Adabas *version*, *x* is the Adabas component (for example, N for nucleus), and *nnnnn* is the Adabas fix number.

Adabas provides a link book containing parameters for invoking the linkage editor for each Adabas phase. The name of each link book begins with "LNK" and the member type is "OBJ".

No link book is provided for module ADAOPD or for any other programs distributed in source form. Programs distributed in source form continue to be modified using assembly and link jobs.

If you choose not to take advantage of the link book facility, remove the INVOLVES LINK= statement from any zap before applying it. You can then run the linkage editor step to recreate the phase separately, as before.

This may be done to link a temporary version of a phase into a separate sublibrary for testing purposes. However, it is also possible to maintain a separate test version of Adabas modules by defining an additional z/VSE system history file. See [Maintaining a Separate Test Environment in z/VSE](#).

### Link Book Update Requirements for Secondary SVC

If you use the link book facility and require a non-standard SVC suffix (for example, if you relink the Adabas 8 SVC to phase ADASVC11), you must remember to update the link book for the SVC (LNKSVC.OBJ) to reflect the new phase name.

The link book provided for ADASVC81 is LNKSVC.OBJ. It contains the following:

```
PHASE ADASVC81,*,NOAUTO,SVA
MODE AMODE(31),RMODE(24)
INCLUDE SVCVSE
INCLUDE SVCCLU
ENTRY ADASVC
```

To set up an SVC with suffix -11, you would need to update the link book as follows:

```
// DLBL SAGLIB,'adabas.Vvrs.library'  
// EXTENT SYS010  
// ASSGN SYS010,DISK,VOL=volser,SHR  
// EXEC LIBR  
ACCESS SUBLIB=SAGLIB.ADAvrs  
CATALOG LNK SVC.OBJ REPLACE=YES  
PHASE ADASVC11,*,NOAUTO,SVA  
MODE AMODE(31),RMODE(24)  
INCLUDE SVCVSE  
INCLUDE SVCCLU  
ENTRY ADASVC  
/+  
/*  
  
—where  
vrs is the Adabas version  
adabas.Vvrs.library is the physical name of the Adabas vrs library  
volser is the volume on which the library resides
```

## Link Book Update Requirements for Running AMODE 24

If you use the link book facility and require AMODE 24 versions of any modules linked by default as AMODE 31 (ADARUN, ADASVC74), you must update the corresponding link book (LNK-RUN.OBJ, LNK SVC.OBJ) to remove the MODE statement.

This link book update can be made using a method similar to that described in the previous section for the SVC suffix update.

## Adabas 8 Adalink Considerations

---

- [Link Routine User Exit 1 \(Pre-Command\) and User Exit 2 \(Post-Command\)](#)
- [LNKUES for Data Conversion](#)
- [ADAUSER Considerations](#)

### 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 an Adalink routine:

- Link routine user exit 1, LUEXIT1, receives control *before* a command is passed to a target with the router 04 call.



**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). LUEXIT1 must check this byte and return if it is set to X'04'. Be sure to reset R15 to zero on return.



- Link routine user exit 2, LUEXIT2, receives control *after* a command has been completely processed by a target, the router, or by the Adalink itself.

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

| Register | Contents   |
|----------|--|
| 1        | Address of the UB.   |
| 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 at xxxxxxxxxx.<br>For batch/TSO, R13 points to the link routine's work area. |
| 14       | Return address   |
| 15       | Entry point address: LUEXIT1 or LUEXIT2  |

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 13.

If at return from LUEXIT1, 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 ACBXRSP to a non-zero value to indicate to the calling program that it has suppressed the command: response code 216 (ADARSP216) is reserved for this purpose.

User buffers can be used to pass data between the link routine user exits 1 and 2 and Adabas nucleus user exits 11 and 4. Refer to *User Buffers* for more information.

An Adalink routine can return the following non-zero response codes in ACBXRSP:

| Response Code   | Description                    |
|-----------------|--------------------------------|
| 213 (ADARSP213) | No ID table                    |
| 216 (ADARSP216) | LUEXIT1 suppressed the command |
| 218 (ADARSP218) | No UB available                |

## LNKUES for Data Conversion

The Adabas 8 standard batch ADALNK is delivered with UES (Universal Encoding Support). The LNKUES module, as well as the modules ASC2EBC and EBC2ASC, are linked into the standard batch ADALNK. LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller.

LNKUES is called only on ADALNK 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.

- For requests, LNKUES receives control before LUEXIT1.
- For replies, LNKUES receives control after LUEXIT2.

By default, two translation tables are linked into LNKUES/ADALNK:

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



**Note:** 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.

If you prefer to use the same translation tables that are used in Entire Net-Work:

- In ASC2EBC and EBC2ASC, change the COPY statements from UES2ASC and UES2EBC to NW2ASC and NW2EBC, respectively.
- Re-assemble the translation tables and re-link LNKUES/ADALNK.

Both the Adabas and Entire Net-Work translation table pairs are provided in the section [Translation Tables](#). You may want to modify the translation tables or create your own translation table pair. Be sure to (re)assemble the translation tables and (re)link LNKUES/ADALNK.

Refer to the member LNKLNK.OBJ for the current link-edit control statements for linking the ADALNK.PHASE. The following is a sample job for (re)linking ADALNK with LNKUES and the translation tables:

```
*
// JOB ...
// EXEC PROC=
// LIBDEF *,SEARCH=(search-chain-library.sublib ...)
// LIBDEF PHASE,CATALOG=(lib.sublib)
  PHASE ADALNK,*
  MODE AMODE(31),RMODE(24)
  INCLUDE LNKVSE8
  INCLUDE LINKIND
  INCLUDE LNGBLS
  INCLUDE LNKUES
  INCLUDE ASC2EBC
  INCLUDE EBC2ASC
  INCLUDE LNKDSL
  INCLUDE RTRVSE
  INCLUDE JNMVSE
  ENTRY ADABAS
// EXEC LNKEDT
```

The (re)linked ADALNK must be made available to Entire Net-Work. If you are calling Adabas 8 and you do not have the correct LNKUES/ADALNK module, Adabas produces unexpected results: response code 022 (ADARSP022), 253 (ADARSP253), etc.

## ADAUUSER Considerations

ADAUUSER is a program that links the user to Adabas. It is specific to an operating system and is independent of release level and mode. It can be used in batch and in some TP environments.

ADAUUSER contains the entry point ADABAS and should be linked with all user programs that call Adabas. No other programs containing the CSECT or entry point name ADABAS can be linked in these load phases.

On the first Adabas call, ADAUSER (CDLOAD) loads the latest version of ADARUN. This makes the calling process release-independent. Subsequent Adabas calls bypass ADARUN.

ADARUN processes its control statements. For the ADARUN setting PROGRAM=USER (the default), ADARUN loads the non-reentrant Adalink modules. To load a reentrant batch link routine, use the ADARUN parameter PROGRAM=RENTUSER. This makes the calling process mode-independent.

## Setting Defaults in ADARUN

The member DEFAULTS.X is available for setting the ADARUN defaults.

DEFAULTS.X uses MSHP CORRECT to install the fix.

The distributed source library contains member ZAPOPT which also contains a zap in MSHP format to set the ADARUN defaults.

| Default Name | Current Value |
|--------------|---------------|
| Device type  | 3390          |
| SVC number   | 45            |
| Database ID  | 1             |



# 5

## Installing Adabas with TP Monitors

---

|  |     |
|--|-----|
| ■ Preparing Adabas Link Routines for z/VSE .....                       | 58  |
| ■ General Considerations for Installing Adabas with CICS .....         | 62  |
| ■ Installing Adabas with CICS under Adabas 8 .....                     | 64  |
| ■ Installing the CICS High-Performance Stub Routine for Adabas 8 ..... | 79  |
| ■ Installing Adabas with Com-plete under Adabas 8 .....                | 95  |
| ■ Installing Adabas with Batch under Adabas 8 .....                    | 97  |
| ■ Establishing Adabas SVC Routing by Adabas Database ID .....          | 99  |
| ■ Modifying Source Member Defaults (LGBLSET Macro) in Version 8 .....  | 108 |

This chapter provides information needed to install Adabas in batch mode and with its teleprocessing (TP) monitors.

## Preparing Adabas Link Routines for z/VSE

---

This section covers the following topics:

- [High-Level Assembler](#)
- [Addressing Mode Assembly Directives](#)
- [UES-Enabled Link Routines](#)



**Important:** If an ADALNK batch link routine has been linked or modified by Software AG 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.

### High-Level Assembler

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

### Addressing Mode Assembly Directives

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

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

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

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

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

### Modifying the Assembly Directives

These directives may be changed by modifying the linkage editor control statements. For example, to link the batch ADALNK module with `AMODE31` and an `RMODE ANY`, the following control statements may be provided as input to the linkage editor:

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

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

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

## Re-linking Adabas 8 Link Routines

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

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

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



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

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

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

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

All of the Adabas 8 CICS link routine modules - `ADACICS`, `ADACICT`, and `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.

## ADAUUSER AMODE/RMODE Considerations

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

## UES-Enabled Link Routines

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

This section covers the following topics:

### Default or Customized Translation Tables

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

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

The two standard translation tables are:

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

The Adabas translation table pair is provided in the section [Translation Tables](#).

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



#### Notes:



1. It should only be necessary to modify these translation tables in the rare case that some country-specific character other than "A-Z a-z 0-9" must be used in the Additions 1 (user ID) or Additions 3 field of the control block.
2. The load module LNKUESL delivered with earlier levels of Adabas Version 7 is no longer supplied since the link jobs now specify the LNKUES or LNKUES7 module and the translation tables separately.
3. The LNKUES module is functionally reentrant; however, they is not linked that way in the Adabas load library.
4. When linking the LNKUES load module and the translation tables, the linkage editor may produce warning messages concerning the reentrant or reusability status of the linked module. These warning messages can be ignored.
5. If relinking an Adabas 8 link routine for UES support, the LNKUES module must be included. This will ensure that your new Adabas 8 applications have support for Adabas 8 direct calls and control blocks.

### Calling LNKUES

LNKUES is called only on Adabas link routine request (X'1C') and reply (X'20') calls if the first byte of the communication ID contains X'01' and the second byte does not have the EBCDIC (X'04') bit set. In Adabas 8 requests, LNKUES receives control before LUEXIT1. In Adabas 8 replies, LNKUES receives control after LUEXIT2.

### Adabas 8 Jobs for z/VSE Universal Encoding Support

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

| Sample Job | Description   |
|------------|---|
| ALNKCIC8.X | Assembles and links the CICS globals table with LNKUES and the default translation tables ASC2EBC and EBC2ASC.                                |
| ALNKLCO8.X | Relinks the Com-plete link routine with the LCOGBL link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.        |
| ALNKLNK8.X | Relinks the batch link routine with the LNKGBLS link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC.           |
| ALNKLNK8.X | Relinks the reentrant batch link routine with the LNKRGBL link globals table, LNKUES, and the default translation tables ASC2EBC and EBC2ASC. |

Before you can use any of these jobs, they should be edited to prepare the job power statements, provide the proper names for the procedures and libraries referenced, and all necessary extent and volume information. Refer to the comments in the jobs themselves for more information.

## Disabling UES Support for Adabas 8 Link Routines

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

### ➤ To disable UES support in link routines:

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

For more information about the specific link routines, read [Installing Adabas with Com-plete under Adabas 8](#), and [Installing Adabas with Batch under Adabas 8](#), elsewhere in this guide.

## General Considerations for Installing Adabas with CICS

---

The Adabas command-level link routine supports the CICS transaction server (TS) 1.1 running under z/VSE 2.4 and above. CICS TS 1.1 running under z/VSE 2.4 and above must run a current version of Adabas and use the command-level link component.



**Note:** The OPID option for the USERID field is no longer supported; therefore, it is not provided with the command-level link routine.

This section covers the following topics:

- [CICS Release Support](#)
- [CICS MRO Environment Requirements](#)
- [Sample Resource Definitions](#)
- [Requirement for CICS Command Resource Security](#)

### CICS Release Support

The Adabas 8 CICS link components are supported for CICS/TS 1.1 and above for z/VSE.

## CICS MRO Environment Requirements

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

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

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

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

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

## Sample Resource Definitions

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

Modify and use the sample DEFINE statements located in member DEFADAC as input to the IBM DFHCSDUP utility to define the Adabas CICS command-level components. Consult the appropriate IBM CICS documentation for information on the DFHCSDUP utility. The DEFADAC member can be found in the Adabas 8 CICS command-level source library (ADA<sub>vrn</sub>.LIBR).

## Requirement for CICS Command Resource Security

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

## Installing Adabas with CICS under Adabas 8

---

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

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

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

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

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

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

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

This section covers the following topics:

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

## ■ Installation Procedure Under Adabas 8

### The Adabas CICS Application Stub (ADACICS)

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

### The Adabas CICS Names Module (ACINAMES)

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

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

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

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

### The Adabas CICS Installation Options Table (ACIOPT)

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

The Adabas CICS installation options table is built using the **MACIOPT macro** (see the MACIOPT macro assembly job, ASMCOPT.X).

## The MACINS Macro

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

The MACINS macro is provided in the Adabas CICS z/VSE sublibrary.

The syntax of the MACINS macro is shown below:

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

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

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

## Example

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

|   |   |   |
|---|---|---|
| * | Sample "ACINAMES" for Adabas multiple-TRUE support.<br>MACINS TRUENAME=ADATRUE,<br>GTNAME=CICSGBL | X |
|---|---|---|

## The MACIOPT Macro

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

The MACIOPT macro is located in the ADA<sub>vrs</sub> sublibrary as member MACIOPT.A on z/VSE systems. A sample ACIOPT source member is provided in the ADA<sub>vrs</sub> sublibrary on z/VSE systems.

The syntax of the MACINS macro is shown below:

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

                        GROUP , GTNAME = link-globals-table-name

FINAL
```

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

- [The ENTRY=GLOBAL Statement](#)
- [The ENTRY=GROUP Statement](#)
- [The ENTRY=FINAL Statement](#)

■ Example

### The ENTRY=GLOBAL Statement

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

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

| Parameter | Description  | Default |
|-----------|--|---------|
| GEN       | Indicates whether the ACIOPT CSECT or a mapping DSECT of the ACIOPT module should be generated.<br><br>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).<br><br>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.<br><br>We recommend that you use the default load module name of ACIOPT, defining ACIOPT to CICS and allowing ADACIC0 to load the ACIOPT module when the program is executed to install the Adabas CICS components.   | ACIOPT  |
| IMSGDEST  | Identifies the destination type for the installation progress and error messages produced by ADACIC0: console, transient data queue, or both.<br><br>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.<br><br>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. | CONSOLE |



| Parameter | Description   | Default              |
|-----------|---|----------------------|
|           | IMSGDEST=BOTH causes installation progress messages to be written both to the console and to a named CICS transient data queue.   |                      |
| IMQNAME   | <p>Specifies the 4-character name of the CICS transient data queue where installation progress and error messages should be written. If IMQNAME is specified then the IMSGDEST parameter must be set to TDQ or BOTH.</p> <p>The named transient data queue must be defined to CICS as either an extra-partition queue or as an indirect queue which references an extra-partition data queue. The simplest way to set up such a data queue is to make it indirect and refer to the CICS-supplied extra-partition data queue CSSL.</p> <p>The queue may be defined using the CICS RDO facility (using the CEDA transaction) or using the DFHDCT macro. On z/VSE systems, the transient data queue must be defined using the DFHDCT macro. A sample member, DCTACIA, is provided in the z/VSE ADA<i>vrs</i> sublibrary. For more information, consult the appropriate IBM CICS documentation.</p> <p>Installation messages written to a CICS transient data queue are variable length records with no printer control character in the first byte of the record. The records will not exceed 132 bytes in length.</p> | There is no default. |
| MNTRUE    | <p>Specifies a maximum value for the number of Adabas CICS execution units (and thus globals tables) to be installed for this CICS or CICSplex.</p> <p>If this number is exceeded, a warning MNOTE and condition code of 4 is produced by the assembler.</p> <p>This parameter is provided as an option to place an upper limit on the number of Adabas CICS execution units that may be installed. You might find this necessary to limit the storage and resource constraints multiple Adabas CICS execution units might place on your system. Although the setting for MNTRUE may be quite high, the storage, resources and Adabas CICS components must be available to be installed.</p>  | 8                    |

### The ENTRY=GROUP Statement

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

Only one parameter can be specified for ENTRY=GROUP:

| Parameter | Description   | Default              |
|-----------|---|----------------------|
| GTNAME    | Specifies the name of the link globals table to be loaded and used to install an Adabas CICS execution unit.<br><br>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
```

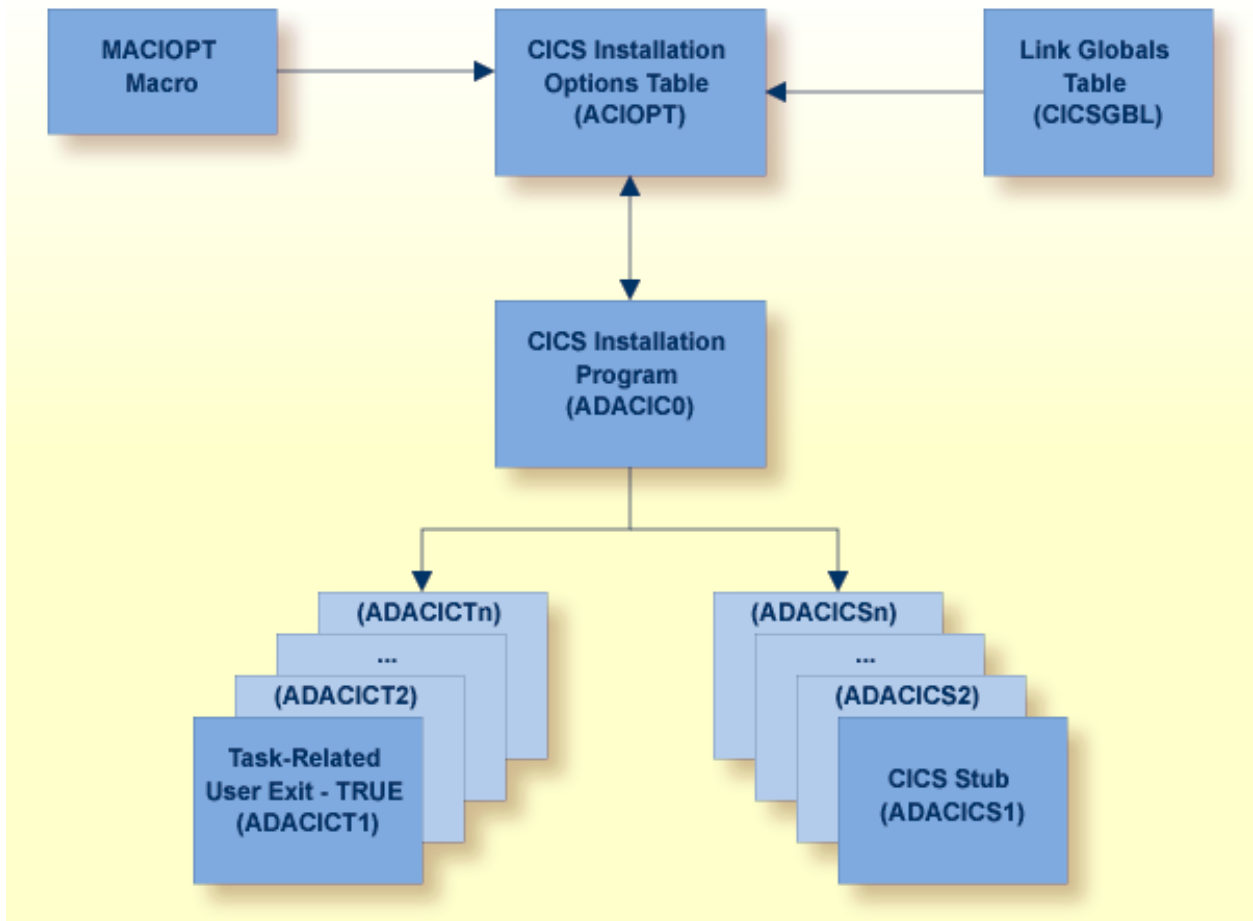
### Adabas Task-Related User Exits (TRUEs)

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.

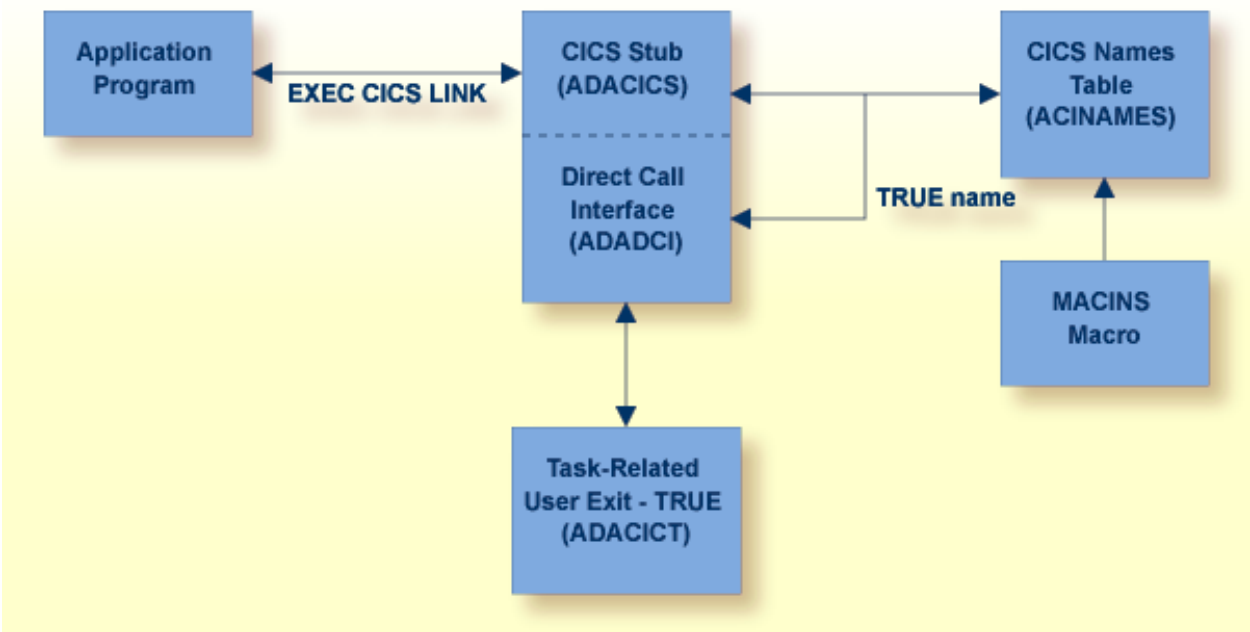
If the Adabas CICS interface employs exits such as the Adabas Fastpath exit or other System Coordinator facilities, there may be more than one invocation of the Adabas TRUE for each EXEC CICS LINK issued by the application program. Other Software AG products that can have multiple TRUE invocations for 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 Modules

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

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

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

| Module        | Description   |
|---------------|---|
| CICSGBL.PHASE | Link globals table executable module.   |
| LNKCIC0.OBJ   | CICS link book used when applying maintenance with MSHP to link-edit ADACIC0.OBJ to produce ADACIC0.PHASE.  |
| LNKCICG.OBJ   | CICS link book used when applying maintenance with MSHP to link-edit the CICSGBL.OBJ globals table and produce CICSGBL.PHASE.                     |
| LNKCICS.OBJ   | CICS link book used when applying maintenance with MSHP to link-edit ADADCI.OBJ and ADACICS.OBJ to produce ADACICS.PHASE.                         |
| LNKCICT.OBJ   | CICS link book used when applying maintenance with MSHP to link-edit ADACICT.OBJ and LNKCIM.OBJ to produce ADACICT.PHASE.                         |
| LNKCIM.OBJ    | CICS task-related user exit (TRUE) product code object module, independent part. This module is linked with ADACICT.OBJ to produce ADACICT.PHASE. |

## Installation Procedure Under Adabas 8

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

- [Step 1. Modify the CICS Startup JCS](#)
- [Step 2. Prepare the Adabas CICS Installation Options Table](#)
- [Step 3. Prepare the Adabas CICS Task-Related User Exits \(TRUEs\) -- ADACICT](#)
- [Step 4. Prepare the Adabas CICS Names Module -- ACINAMES](#)
- [Step 5. Prepare the Adabas CICS Application Stub -- ADACICS](#)
- [Step 6. Prepare the CICS Link Globals Table -- CICSGBL.A](#)
- [Step 7. Assemble and Link-edit the CICS Link Globals Table \(ALNKCIC8.X\)](#)
- [Step 8. Modify CICS Installation Values \(DEFADAC.A\)](#)
- [Step 9. Update the CICS CSD File \(DFHCSDUP\)](#)
- [Step 10. Modify, Assemble and Link the CICS PLTPI Table for ADACIC0](#)
- [Step 11. Update, Assemble and Link-edit the Destination Control Table \(DCTACI.A\)](#)
- [Step 12. Start the CICS](#)

### Step 1. Modify the CICS Startup JCS

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

## Step 2. Prepare the Adabas CICS Installation Options Table

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

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

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

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

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

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

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

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

The MACIOPT macro is located in the ADA<sub>vrs</sub> sublibrary as member MACIOPT.A on z/VSE systems. For complete information on the MACIOPT macro, read [The MACIOPT Macro](#), elsewhere in this section.

- 2 Assemble and link the ACIOPT source module either as the standalone module named "ACIOPT" or with any load module name linked with ADACIC0. If linked as a standalone module it must be named "ACIOPT" and it must be defined as a program to CICS.

The ACIOPT module may be defined to CICS using the CEDA/RDO facility or the DFHCSDUP utility. Sample DFHCSDUP statements are provided in the DEFADAC member in the ADA<sub>vrs</sub> sublibrary on z/VSE systems.

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

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



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

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

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

#### Example

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

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

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

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

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

- Code the source for the ACINAMES module using the MACINS macro. For complete information, read [The MACINS Macro](#), elsewhere in this section.

The MACINS macro is provided in the Adabas CICS z/VSE sublibrary.

### Example

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

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

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

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

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

A sample job, ASMCINS.X , is provided.

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

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

### Example

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

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



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

#### Step 6. Prepare the CICS Link Globals Table -- CICSGBL.A)

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

Modify the sample CICSGBL.A member found in the Adabas 8 ADA<sub>vrs</sub> sublibrary. This member contains sample default installation (LGBLSET) parameter settings. For more information about what to modify in this member, read [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), elsewhere in this section.



#### Notes:

1. Adabas no longer supports the ADACIRQ module or the reading of an input CICS transient data queue to obtain the name of the link globals table during installation. This was necessary to permit the installation of multiple Adabas CICS execution units from the same installation program.
2. The setting for the OPSYS parameter must be set to "VSE".

#### ➤ To prepare the link globals table, complete the following steps:

- 1 Code the link globals table using the LGBLSET macro as described in [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), elsewhere in this section.

The OPSYS parameter must be set to "VSE".

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

- 2 Save the modified CICSGBL.A member with a unique name in an appropriate user sublibrary.

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

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

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

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

**Step 9. Update the CICS CSD File (DFHCSDUP)**

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

**Step 10. Modify, Assemble and Link the CICS PLTPI Table for ADACIC0**

Modify the CICS PLTPI table to add an entry for the CICS installation program ADACIC0. The ADACIC0 installation program will start the TRUEs once CICS is started. Use member ADAPLTXX from the Adabas 8 ADA<sub>vrn</sub>.LIBR library as a sample for enabling and starting a legacy Adabas TRUE and the new Version 8 TRUE in the second phase of the PLT.

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

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

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

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

## Step 12. Start the CICS

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

## Installing the CICS High-Performance Stub Routine for Adabas 8

---

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 command-level link component to applications written in languages other than Software AG's Natural (for example, Assembler, COBOL, PL/I).



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

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

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

A CICS/TS 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. CICS Command-Level Link Required

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

3. Supported Programming Languages

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

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

## Stub Components

| Type        | Member  | Description                                       |
|-------------|---------|---|
| Source      | ADAGSET | macro required for assembling LNCSTUB and ALCSIVP |
|             | ALCSIVP | source for the ALC install verification           |
|             | COBSIVP | source for the COBOL install verification         |
|             | LNCSTUB | source for the high-performance stub              |
| Job control | JCLALCI | sample JCL for ALC install verification           |
|             | JCLCOBI | sample JCL for COBOL install verification         |
|             | JCLLNCS | 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<sub>vr</sub>s.SRCE library.

Step 1 has the following substeps:

- [Edit the ADAGSET Macro](#)
- [\(Optional\) Set the LNCSTUB Entry-Point Alias](#)
- [Modify Member JCLLNCS](#)
- [Preprocess, Assemble, and Link the LNCSTUB Module](#)

- [Make the LNCSTUB Available to Application Programs](#)

### Edit the ADAGSET Macro



**Note:** For information about editing the ADAGSET macro, refer to the section [Modifying Source Member Defaults \(ADAGSET Macro\)](#), 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 CICS link routine or CICS stub to be invoked by the LNCSTUB and ALCSIVP programs. If your Adabas CICS command-level link component program has been linked with a name other than ADACICS, change the value of the ENTPT keyword in the [ADAGSET macro](#). The value in this field is used in the priming EXEC CICS LINK command issued by LNCSTUB.

### (Optional) Set the LNCSTUB Entry-Point Alias

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

### Modify Member JLLNCS

Member JLLNCS (in the ADA<sub>vr</sub>s.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.                                |

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

### Preprocess, Assemble, and Link the LNCSTUB Module

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



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

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

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

```
NAME LNCSTUB(R)
```

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

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

## Make the LNCSTUB Available to Application Programs

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

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

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



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

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

## Step 2: (Optional) Install and Execute an IVP

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

This section describes each of these IVPs:

- [Install and Execute the Assembler IVP: ALCSIVP](#)
- [Install and Execute the COBOL IVP: COBSIVP](#)



**Note:** The two installation verification programs ALCSIVP and COBSIVP only use fields AA and AE from the Software AG-provided demonstration EMPLOYEES file. For more information about the Software AG-provided demonstration files, read [Load the Demonstration Files](#) in the z/OS installation instructions, 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<sub>vr</sub>s.SRCE:

- Edit the file number field DBFNR to be sure it matches the value needed to access the EMPLOYEES file on the Software AG-provided demonstration database you intend to use. For more information about the Software AG-provided demonstration files, read [Load the Demonstration Files](#) in the z/OS installation instructions, 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 command-level link component program. The field LNCNAME is now used and it derives its value from the ENTPT keyword of the ADAGSET macro.

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



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

#### 2 Modify the sample job stream, JCLALCI in ADA<sub>vr</sub>s.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) DATALLOCATION(ANY)
EXECKEY(USER) EXECUTIONSET(FULLAPI)

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

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

### Install and Execute the COBOL IVP: COBSIVP

Member COBSIVP illustrates the use of the Adabas DCI with a COBOL program. COBSIVP 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*vars*.SRCE:
  - Edit the fields WORK-DBID and WORK-FNR to place the desired database ID and file number in the VALUE clauses to access the EMPLOYEES file on the Software AG-provided demonstration database you intend to use. For more information about the Software AG-provided demonstration files, read [Load the Demonstration Files](#) in the z/OS installation instructions, 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 ADA *vrs*.JOBS:

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

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

## 3 Preprocess, compile, and link COBSIVP:

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

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

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

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



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

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

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

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

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

### Step 3: Link and Execute the Application Program

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

Step 3 has the following substeps:

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

## Guidelines for Modifying the Application Program

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

### ■ Programming Languages Supported by LNCSTUB

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

### ■ Use of the CICS Transaction Work Area

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

### ■ Use of the CICS COMMAREA

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

### ■ Reentrant Requirement

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

### ■ CICS Requests Issued by LNCSTUB

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

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

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

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

### ■ DCI Entry Point Address

An EXEC CICS LINK command is issued by LNCSTUB at least once to acquire the DCI entry point from the Adabas CICS command-level link component program. This address is then used for BALR access on all subsequent Adabas calls for a transaction. Thus, the calling applic-

ation program must provide a fullword (4-byte) field to hold the DCI entry point address obtained by LNCSTUB. This 4-byte field is the first parameter passed to the LNCSTUB module by the call mechanism. The remaining parameters comprise the Adabas parameter list needed to execute an Adabas request. (Either a version 7 or version 8 parameter list may be used)

#### ■ DCI Parameter List

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

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

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

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

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

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

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

```
DFHEISTG DSECT
*
PARMLIST DS OF
DS A(DCIPTR)
DS A(ADACB)
DS A(ADAFB)
DS A(ADARB)
DS A(ADASB)
```

```

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

```



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

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

```

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

```

### ■ Restrictions on Application Program Coding

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

### ■ Standard Batch Call Mechanism Used

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

## Link the Application Programs to Include the LNCSTUB Module

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

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

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

### ➤ For linking Assembler language programs:

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

```
INCLUDE DFHEAI
```

The Assembler object input is similar to:

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

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

### ➤ For linking COBOL language programs:

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

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

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

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

**Modifying Source Member Defaults (ADAGSET Macro)**

 **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, Software AG recommends that you program the ADAGSET macro with site-specific default values and put it in a source library that is available in the SYSLIB concatenation during assembly.

The 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 Command-Level Link Routine
- LOGID: Default Logical Database ID
- PARMTYP: Area for Adabas Parameter List

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

| Parameter | Description  | Syntax                                |
|-----------|--|---------------------------------------|
| ENTPT     | The name of the Adabas CICS Command-Level link routine used in the priming EXEC CICS LINK command issued by LNCSTUB. | ENTPT={ ' <u>ADACICS</u> '   'name' } |

**LOGID: Default Logical Database ID**

| Parameter | Description  | Syntax            |
|-----------|--|-------------------|
| LOGID     | The database ID used by the (optional) Assembler IVP, ALCSIVP..<br><br>Valid ID numbers are 1-65535. | LOGID= <i>nnn</i> |

**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.<br><br>This value should match that specified for the LGBLSET parameter of the same name in use by the Adabas CICS Command-level link routine. | PARMTYP={ <u>ALL</u>   COM ↔<br>  TWA } |

## Installing Adabas with Com-plete under Adabas 8

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



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

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

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

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

System Server running under Com-plete. At this time, Com-plete does not support a mixed Adabas 7 and Adabas 8 link routine environment; thus Com-plete must be run with either an Adabas 7 link routine or an Adabas 8 link routine.

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

➤ **To prepare the Adabas 8 link routine:**

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



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

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

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

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



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

The Adabas 8 link routine is prepared.

## Installing Adabas with Batch under Adabas 8

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

Batch applications should be linked with the ADAUSER module to provide the greatest degree of application calling isolation when invoking the Adabas batch link routines. The ADAUSER module will provide code to load the appropriate link routine and the supporting ADARUN and ADAIOR modules. ADARUN, in turn, loads other modules. To start a user program linked with ADAUSER, the following modules must be available in the LIBDEF search chain: ADAIOR, ADAIOS, ADALNK, ADAMLF, ADAOPD, ADAPRF, and ADARUN. In addition, ADAUSER reads DDCARD input from SYSIPT or DISK to allow jobstep setting of the database ID, Adabas SVC number, and other parameters.

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

If you want to use reentrant batch operations, the ADAUSER module can still be linked with the application program, but the PROG=RENTUSER keyword must be coded on the DDCARD input. ADAUSER is, however, non-reentrant. For full reentrant batch applications, it will either need to be loaded (CDLOAD) separately, or the ADALNKR.PHASE must be loaded without using the ADAUSER module. In this case, the default values for DBID, SVC number, length of user information, and which exits are to be used is provided by the linked link globals table, as modified (read [Installing the Reentrant Batch z/VSE Adabas 8 Link Routine](#), elsewhere in this section. It is also possible to zap the ADALNKR.PHASE or LNKVSR8.OBJ module with these defaults, but Software AG recommends coding and linking the link globals table instead. Additional information on using a reentrant batch link routine is also provided in *Required Application Reentrancy Properties* in *&adamf\_op*.

This section covers the following topics:

- [Supplied Modules](#)
- [Installing the Batch z/VSE Adabas 8 Link Routine](#)
- [Installing the Reentrant Batch z/VSE Adabas 8 Link Routine](#)



**Important:** If an ADALNK batch link routine has been linked or modified by Software AG 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 8 with batch.



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

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

## Installing the Batch z/VSE Adabas 8 Link Routine

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

- 1 Edit member LNKGBLS.A in the Adabas distribution sublibrary. Provide values for the LOGID, SVC, GBLNAME, and other keywords to suit your installation requirements. This module contains LGBLSET parameters used to create default settings for link components. A complete description of LGBLSET parameters can be found in [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), elsewhere in this guide.



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

- 2 Edit the ALNKLNK8.X member found in the Adabas 8 sublibrary. This member will assemble and catalog the LNKGBLS.A module and link it and any desired exits with the LNKVSE8.OBJ module to create the ADALNK.PHASE member for Adabas 8. The ALNKLNK8.X member includes sample link-edit control cards to support UES by including the LNKUES.OBJ. module with the ASC2EBC and EBC2ASC translation tables. Modify the link-edit control cards to include any additional Software AG exit or user exit, as specified in the updated LNKGBLS.A member.
- 3 Provide the ADALNK.PHASE member in the LIBDEF search chain for the jobstep that will require Adabas database access or Software AG services.

## Installing the Reentrant Batch z/VSE Adabas 8 Link Routine

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

- 1 Edit member LNKRGBL.A in the Adabas distribution sublibrary. Provide values for the LOGID, SVC, GBLNAME, and other keywords to suit your installation requirements. This module contains LGBLSET parameters used to create default settings for link components. A complete description of LGBLSET parameters can be found in [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), elsewhere in this guide.



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

- 2 Edit the ALNKLNK8.X member found in the Adabas 8 sublibrary. This member will assemble and catalog the LNKRGBL.A module and link it and any desired exits with the LNKVSE8.OBJ module to create the ADALNKR.PHASE member for Adabas 8. The ALNKLNK8.X member includes sample link-edit control cards to support UES by including the LNKUES.OBJ. module with the ASC2EBC and EBC2ASC translation tables. Modify the link-edit control cards to include any additional Software AG exit or user exit, as specified in the updated LNKRGBL.A member.
- 3 Provide the ADALNKR.PHASE member in the LIBDEF search chain for the jobstep that will require Adabas database access or Software AG services.

## Establishing Adabas SVC Routing by Adabas Database ID

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

Duplicate database IDs are not allowed in the DBID/SVC routing table as there is no reliable way for the link routine to determine which SVC should be used for a database ID if it is listed more



than once. If duplicate database IDs are found while the table is being assembled, they are flagged with an assembler MNOTE and a return code of 16 is returned for the assembly attempt.

**Notes:**

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



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

This section covers the following topics:

- [Installing the Adabas DBID/SVC Routing Feature](#)
- [General Operation](#)
- [Using the MDBSVC Macro](#)

## Installing the Adabas DBID/SVC Routing Feature

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

1. Define the DBID/SVC routing table in a library member using MDBSVC macro statements. For more information about the DBID/SVC routing table and the MDBSVC macro, read [Using the MDBSVC Macro](#), elsewhere in this section.
2. Assemble and link-edit the DBID/SVC routing table member to create a load module or PHASE that will be made available to the operating environment where the SVC routing feature will be used.
3. Modify a link globals table for the operating environment, specifying the LGBLSET keywords DYNDBSVC=YES and DBSVCTN=*name*, where *name* is the name of the DBID/SVC routing table load module that should be used by the link routine. Assemble and link-edit the updated link globals table as required for the operating environment. For more information about the link globals table and the LGBLSET macro, read [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), 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 z/OS or VSE TP monitoring environment, provided elsewhere in this section.

4. Make the prepared DBID/SVC routing table available in a load library that is accessible by the application program's job step, so it can be loaded by the link routine when it runs.
5. Except for CICS systems, you will need to relink ADALNK or ADALNKR making sure that the INCLUDE statements for the LNKDSL and DEPRTR (or RTRVSE on VSE) modules are included in the job.

This section covers the following topics:

- [Installing DBID/SVC Routing under z/VSE Batch](#)
- [Installing DBID/SVC Routing under CICS](#)

### Installing DBID/SVC Routing under z/VSE Batch

» To install the Adabas DBID/SVC routing feature under z/VSE batch, complete the following steps:

- 1 Define or modify the DBID/SVC routing table by coding a series of MDBCSVC macros in a library member. Sample member ADASVCTB.A is provided in the sublibrary SAGLIB.ADA<sub>vrs</sub> 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 PHASE that you can make available to the application execution job step. The PHASE should be linked non-reusable and non-reentrant because the link routine subprogram LNKDSL will need to store the addresses of the Adabas SVC IDT headers in the DBID/SVC module to reduce the operating overhead on multiple commands accessing the same Adabas SVC.
- 3 Define or modify a link globals table for the execution environment. The following LGBLSET keywords are required to support the Adabas SVC routing feature:

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

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

| Link Routine  | Sample Job |
|---------------|------------|
| ADALNK.PHASE  | ALNKLNK8.X |
| ADALNKR.PHASE | ALNKLNK8.X |

## Installing DBID/SVC Routing under CICS

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

- 1 Define or modify the DBID/SVC routing table by coding a series of MDBCSVC macros in a library member. Sample member ADASVCTB is provided in the ADA<sub>vr</sub>s.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<sub>vr</sub>s.SRCE library for sample DFHCSDUP definition statements. The program attributes should be Reload(No), Resident(Yes), Dataloc(Any), and Exekey(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= <i>name</i>    | This keyword specifies the name of the DBID/SVC table for this job step. This name must match the name of the load module created to ensure the proper table is loaded when the link routine runs. |

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

## General Operation

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

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

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

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

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

## Using the MDBSVC Macro

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

```

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

```

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

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

| MDBSVC Statement Type | Description   | Number Allowed         |
|-----------------------|---|------------------------|
| TYPE=INIT             | Only one MDBSVC TYPE=INIT statement can be included in the DBID/SVC routing table source member and it must be the first MDBSVC statement in the member. This statement identifies the beginning of the DBID/SVC routing table. The MDBSVC TYPE=INIT statement 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={ZQS|VSE}]
```

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. In z/OS environments, database IDs must range from 1 to 65535. The same database ID cannot be specified on multiple MDBSVC statements; they must be unique across all of the DBID and DBID2 statements in the DBID/SVC routing table. Duplicate

values are flagged with an MNOTE, which causes the assembly of the DBID/SVC routing table to stop with return code 16.

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

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

## DBID2

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

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

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

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

## OPSYS

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

## PREFIX

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

## SVC

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

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

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

## TABNAME

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

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

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

## Modifying Source Member Defaults (LGBLSET Macro) in Version 8

---

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

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

- ADL: Adabas Bridge for DL/I Support
- AVB: Adabas Bridge for VSAM Support
- CITSNM: Adabas CICS TS Queue Name
- COR: SYSCOR Exit Support
- DBSVCTN: DBID/SVC Routing Table
- DYNDBSVC: DBID/SVC Routing Table
- ENTPT: Name of the Adabas CICS Command-Level Link Routine
- GBLNAME: Name of Link Globals Module
- GEN: Generate CSECT or DSECT
- IDTNAME: BS2000 IDT Common Memory Name
- IDTUGRP: BS2000 Memory Pool User Bound
- LOGID: Default Logical Database ID
- LUINFO: Length of User Data Passed to Adabas LUEXIT1 and LUEXIT2
- LUSAVE: Size of User Save Area for Adabas LUEXIT1 and LUEXIT2
- LX1NAME: User Exit 1 Module Name
- LX2NAME: User Exit 2 Module Name
- MRO: Multiple Region Option
- NETOPT: Method Used to Create User ID
- NTGPID: Natural Group ID
- NUBS: Number of User Blocks Created By CICS Link Routine
- OPSYS: Operating System
- PARMTYP: Area for Adabas Parameter List
- PRE: DSECT Data Prefix
- PURGE: Purge Transaction
- RENT: Reentrant Module Flag
- RETRYX: Retry Command Exit Flag
- REVHID: Adabas Review Hub ID Support
- REVIEW: Adabas Review Support
- REVREL: Adabas Review Release
- RMI: Resource Manager Interface
- 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
- TPMON: Operating Environment
- TRUENM: CICS TRUE Name
- UBPLC: User Block Pool Allocation
- UBSTIME: User Block Scan Time
- UBTYPE: User Block Type
- UES: Universal Encoding Support
- USERX1: User Exit 1 Flag
- USERX2: User Exit 2 Flag
- XWAIT: XWAIT Setting for CICS

### ADL: Adabas Bridge for DL/I Support

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

### AVB: Adabas Bridge for VSAM Support

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

**COR: SYSCOR Exit Support**

| Parameter | Description   | Syntax         |
|-----------|---|----------------|
| COR       | <p>Indicates whether or not the Adabas System Coordinator (COR) exit is installed and active (as required by Adabas Fastpath, Adabas Vista, and Adabas Transaction Manager).</p> <ul style="list-style-type: none"> <li>■ COR=YES: The COR exit is installed and active.</li> <li>■ COR=NO: The COR exit is <i>not</i> installed and active.</li> </ul> | COR={NO   YES} |

**DBSVCTN: DBID/SVC Routing Table**

| Parameter | Description  | Syntax                             |
|-----------|--|------------------------------------|
| DBSVCTN   | <p>Provides the name of the DBID/SVC routing table that should be used by the link routine during its execution, if any.</p> <p>The routing table name must conform to names for z/OS standard load modules. It is used by a z/OS LOAD macro/SVC during batch, TSO, or IMS operation or by an EXEC CICS LOAD PROGRAM command during CICS operation.</p> <p>If the load module listed is not found, or if it is found to contain invalid header information, user abend U657 is issued in batch, TSO, or IMS environments.</p> <p>If the load module is not defined to CICS or not found in the CICS DFHRPL concatenation, the Adabas CICS link routine environment is not initialized.</p> <p><b>Note:</b> If the DYNDBSVC parameter is set to NO, this parameter setting is ignored.</p> <p>For more information about SVC routing by database ID in z/OS environments, read <i>Establishing Adabas SVC Routing by Adabas Database ID</i>, in the <i>Adabas z/OS Installation Guide</i> documentation.</p> <p><b>Note:</b> 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</p> | DBSVCTN={ <i>name</i>   ADASVCTB } |

| Parameter | Description   | Syntax |
|-----------|---|--------|
|           | router for any given session, not across routers. To avoid problems if the dynamic SVC by DBID routing feature is enabled for these products, error messages are issued, the assembly step of the globals table will receive return code 16, and the globals table load module will not be generated. |        |

### 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.<br><br>For more information about SVC routing by database ID in z/OS environments, read <i>Establishing Adabas SVC Routing by Adabas Database ID</i> , in the <i>Adabas z/OS Installation Guide</i> 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.<br><br>See also notes 1 and 2 in the installation procedure. | ENTPT={ <u>ADACICS</u>   name } |

### GBLNAME: Name of Link Globals Module

| Parameter | Description                          | Syntax                            |
|-----------|--------------------------------------|-----------------------------------|
| GBLNAME   | The name of the link globals module. | GBLNAME={ <u>LNKGBLS</u>   name } |

### GEN: Generate CSECT or DSECT

| Parameter | Description                                      | Syntax                       |
|-----------|--|------------------------------|
| GEN       | Indicates whether a CSECT or DSECT is generated. | GEN={ <u>CSECT</u>   DSECT } |

**IDTNAME: BS2000 IDT Common Memory Name**

| Parameter | Description                                    | Syntax               |
|-----------|--|----------------------|
| IDTNAME   | The common memory pool name of the BS2000 IDT. | IDTNAME= <i>name</i> |

**IDTUGRP: BS2000 Memory Pool User Bound**

| Parameter | Description   | Syntax                      |
|-----------|---|-----------------------------|
| IDTUGRP   | Indicates whether the common memory pool is user bound (BS2000) | IDTUGRP={ <u>NO</u>   YES } |

**LOGID: Default Logical Database ID**

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

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

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

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

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

**LX1NAME: User Exit 1 Module Name**

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

**LX2NAME: User Exit 2 Module Name**

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

**MRO: Multiple Region Option**

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

**NETOPT: Method Used to Create User ID**

| Parameter | Description  | Syntax                     |
|-----------|--|----------------------------|
| NETOPT    | <p>If NETOPT=YES is specified, an 8-byte user ID will be constructed from the VTAM LU name. If NETOPT=NO is specified, the user ID is created from the constant "CICS" plus the four-byte CICS terminal ID (TCTTETI) for terminal tasks. For non-terminal tasks, the user ID comprises the constant "C" plus the CICS task number, in zoned decimal format, including leading zeroes.</p> <p>If you run with the CICS multiple region option (MRO), you must use the default value for this option. If NETOPT=YES and MRO=YES are specified, an assembler MNOTE and a return code of 16 are produced from the assembly step.</p> | NETOPT={ <u>NO</u>   YES } |

**NTGPID: Natural Group ID**

| Parameter | Description  | Syntax                      |
|-----------|--|-----------------------------|
| NTGPID    | <p>Specifies a four-byte Natural group ID as required for unique Adabas user ID generation in the CICS sysplex environment with Natural Version 2.2 SP8 and above. The value is associated with all users who call the Adabas command-level link routine assembled with the specified value.</p> <p>There is no default value. If no value is specified, the Adabas internal user ID is built in the conventional manner.</p> <p>Any four-byte alphanumeric value may be specified, but it must be unique for each Adabas command-level link routine running in a CICS sysplex, or z/OS image. If more than one NTGPID is required (for example, both test and production Natural 2.2 SP8), more than one Adabas command-level link routine with associated TRUE must be generated.</p> <p>If you run with the CICS multiple region option (MRO), you may use NTGPID to provide a 4-byte literal for the Adabas communication ID to be used by the Adabas SVC when multiple application regions call Adabas.</p> | NTGPID= <i>4-byte-value</i> |

**NUBS: Number of User Blocks Created By CICS Link Routine**

| Parameter | Description  | Syntax                              |
|-----------|--|-------------------------------------|
| NUBS      | <p>The number of user blocks (UBs) to be created in the user block pool by the CICS link routine. The number of blocks must be large enough to handle the maximum possible number of concurrent Adabas requests.</p> <p><b>Note:</b> The Adabas 6.2 and above command-level link routine obtains storage for the user blocks (the UB pool) above the 16-megabyte line.</p> | NUBS={ <u>100</u>   <i>blocks</i> } |

**OPSYS: Operating System**

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

**PARMTYP: Area for Adabas Parameter List**

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

**PRE: DSECT Data Prefix**

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

**PURGE: Purge Transaction**

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

**RENT: Reentrant Module Flag**

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

**RETRYX: Retry Command Exit Flag**

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

**REVHID: Adabas Review Hub ID Support**

| Parameter | Description   | Syntax               |
|-----------|---|----------------------|
| REVHID    | <p>Specifies the preferred Adabas Review hub ID. This value can be checked during the Adabas TP monitoring installation or during the monitor activate process.</p> <p>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.</p> <p>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:</p> <div>REVHID requires<br/>REVIEW=YES</div> <p>This parameter is not valid in VSE environments. The parameter exists in VSE environments, but should be set to "0".</p> | REVHID= <i>hubid</i> |

**REVIEW: Adabas Review Support**

| Parameter | Description   | Syntax                     |
|-----------|---|----------------------------|
| REVIEW    | Indicates whether or not Software AG's Adabas Review performance monitor is installed and active. | REVIEW={ <u>NO</u>   YES } |



**REVREL: Adabas Review Release**

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

**RMI: Resource Manager Interface**

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

**RTXNAME: Command Retry Exit Name**

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

**RVCLNT: Adabas Review Client Reporting Allowance Setting**

| Parameter | Description   | Syntax                     |
|-----------|---|----------------------------|
| RVCLNT    | This parameter is not valid in VSE environments. The parameter exists in VSE environments, but should be set to "NO". | RVCLNT={ <u>NO</u>   YES } |

**SAF: Adabas Security Interface Flag**

| Parameter | Description  | Syntax                   |
|-----------|--|--------------------------|
| SAF       | <p>Indicates whether Software AG's Adabas SAF Security support is required.</p> <p>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.</p> | SAF={ <u>N</u> O   YES } |

**SAP: SAP Application Support**

| Parameter | Description  | Syntax                   |
|-----------|--|--------------------------|
| SAP       | <p>Indicates whether or not SAP user ID generation is supported.</p> <p>If SAP=YES is specified, the program will detect a SAP initialization call and set the user ID for SAP applications from the constant provided on the initialization call, plus the field ACBADD2.</p> <p>For more information, refer to the supplementary information provided to customers using the SAP application system.</p> | SAP={ <u>N</u> O   YES } |

**SAPSTR: SAP ID String**

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

**SVCNO: Adabas SVC number**

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

## TPMON: Operating Environment

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

## TRUENM: CICS TRUE Name

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

## UBPLOC: User Block Pool Allocation

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

**UBSTIME: User Block Scan Time**

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

**UBTYPE: User Block Type**

| Parameter | Description   | Syntax                        |
|-----------|---|-------------------------------|
| UBTYPE    | <p>Identifies the kind of user block (UB) storage the Adabas CICS installation program and Adabas task-related user exit (TRUE) should obtain and use.</p> <p>Valid values are TASK and POOL. POOL is the default. UBTYPE=POOL causes the installation program to obtain a pool of user blocks in CICS storage. This is the classic mechanism used by Adabas CICS link routines.</p> <p>UBTYPE=TASK changes the behavior of the Adabas CICS installation program and Adabas TRUE so they obtain a single user block element, including any required extensions for user data and Software AG products, for each CICS task that invokes the Adabas TRUE. The user block is obtained in CICS shared storage in user-key. It is released when the Adabas TRUE is driven by CICS at the end of the CICS task. The advantage of UBTYPE=TASK is that there is no scan time required to locate and lock a given UB pool element on each Adabas call. The disadvantages of using UBTYPE=TASK are that a CICS GETMAIN must</p> | UBTYPE={ <u>POOL</u>   TASK } |

| Parameter | Description   | Syntax |
|-----------|---|--------|
|           | <p>be issued for each CICS task the first time the Adabas TRUE is invoked for the task and that a CICS FREEMAIN must be issued to release the user block storage at the end of the CICS task.</p> <p>UBTYPE=POOL should be used if any of the following are true:</p> <ul style="list-style-type: none"> <li>■ The majority of Adabas CICS transactions are short running tasks issuing a relatively small number of Adabas calls per CICS task.</li> <li>■ The CICS system is subject to CICS storage fragmentation.</li> <li>■ Applications running the zIIP Enabler for Natural are present in this CICS system.</li> </ul> <p>Otherwise, UBTYPE=TASK may be considered if:</p> <ul style="list-style-type: none"> <li>■ The majority of Adabas CICS transactions are long running tasks issuing a relatively large number of Adabas calls per CICS task.</li> <li>■ CICS tasks frequently trip CPU limits set by CICS execution monitoring programs such as those from Omegamon.</li> </ul> |        |

### UES: Universal Encoding Support

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

### USERX1: User Exit 1 Flag

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

### USERX2: User Exit 2 Flag

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

**XWAIT: XWAIT Setting for CICS**

| Parameter | Description  | Syntax         |
|-----------|--|----------------|
| XWAIT     | <p>Indicates whether a standard EXEC CICS WAITCICS (XWAIT=NO) or a WAIT EVENTS EXTERNAL (XWAIT=YES) will be executed by the Adabas task-related user exit (TRUE). XWAIT=YES is the default.</p> <p>The CICS WAIT EVENTS EXTERNAL (XWAIT=YES) is the recommended interface for all supported versions of CICS/TS.</p> <p>The CICS WAITCICS statement (XWAIT=NO) is provided but may result in poor CICS transaction performance or unpredictable transaction results in busy CICS environments.</p> | 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.

# 6      Enabling Universal Encoding Support (UES) for Your Adabas Nucleus

---

|   |     |
|---|-----|
| ■ Installing UES Support for the Adabas Nucleus ..... | 125 |
|---|-----|

Prior to Adabas Version 7, Entire Net-Work converted all data for mainframe Adabas when necessary from ASCII to EBCDIC. Starting with Version 7, 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.

For information about UES, 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*.

A sample startup job for a UES-enabled nucleus is provided in member ADANUCU of the ADA<sub>VRS</sub>.JOBS data set. For more information, read *JCL Required for UES Support (z/VSE)*, in the *Adabas Operations Manual*.

2. Ensure that UES support has been activated in the Adabas link routines. Verify that the load modules for all Adabas 8 link routines have been linked with LNKUES and the default (or updated) translation tables and that the LGBLSET **SVCNO parameter** has been set. For Adabas 8, UES is enabled by default for *all* link routines. 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, Software AG internal product software (APS), or through Entire Net-Work (WCP). Connections through Com-plete or Software AG internal product software (APS) use the Adabas Com-plete link routines; connections through Entire Net-Work use the Adabas batch link routines.

Effective with Adabas Version 8.3, only APS version 3.3.1 fix pack 19 and above are supported on z/VSE. Users should note that, with APS version 3.3.1 fix pack 19, a new system parameter TCPIP has been introduced. Users should add parameter TCPIP=NO to the APS reader files in use for all UES-enabled Adabas nucleus jobs.



**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.



## Installing UES Support for the Adabas Nucleus

The following LIBR sublibraries are distributed with Adabas for UES support:

```
SAGLIB.ADAvrsCS
```

```
SAGLIB.APSvrsnn
```

```
SAGLIB.APSvrs
```

### ➤ To install these libraries:

- 1 Create a z/VSE sublibrary for the code pages.

```
* $$ JOB JNM=CRUESL,CLASS=0,DISP=D,LDEST=(,xxxxxx)
* $$ LST CLASS=A,DISP=D
// JOB LIBRDEF
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,'SAG.ADABAS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// EXEC LIBR
DEFINE L=DDECSOJ R=Y
/*
/&
* $$ EOJ
```

- 2 Create a z/VSE sublibrary for APS.

```
* $$ JOB JNM=CRUAPS,CLASS=0,DISP=D,LDEST=(,xxxxxx)
* $$ LST CLASS=A,DISP=D
// JOB LIBRDEF
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,'SAG.APS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// EXEC LIBR
DEFINE L=APSVrsnn R=Y
DEFINE L=APSVrs      R=Y
/*
/&
* $$ EOJ ↵
```

- 3 Restore the UES code pages sublibrary to this file. Refer to the *Software AG Product Delivery Report* for the file positions on the distribution tape.

```
* $$ JOB JNM=RESTECS,DISP=D,CLASS=0,LDEST=(,xxxxxx)
* $$ LST DISP=D,CLASS=A
// JOB RESTECS
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL ADALIB,'SAG.ADABAS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// ASSGN SYS006,cuu
// MTC REW,SYS006
// EXEC LIBR,PARM='MSHP'
RESTORE SUBLIB=SAGLIB.ADAvrsCS:ADALIB.DDECSOJ -
TAPE=SYS006 -
LIST=YES -
REPLACE=YES
/*
// MTC REW,SYS006
// ASSGN SYS006,UA
/&
* $$ EOJ
```

- 4 Restore the APS sublibraries. Refer to the *Software AG Product Delivery Report* for the file positions on the distribution tape.

```
* $$ JOB JNM=RESTAPS,DISP=D,CLASS=0,LDEST=(,xxxxxx)
* $$ LST DISP=D,CLASS=A
// JOB RESTAPS
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL APSLIB,'SAG.APS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// ASSGN SYS006,cuu
// MTC REW,SYS006
// EXEC LIBR,PARM='MSHP'
RESTORE SUBLIB=SAGLIB.APSvrsnn:ADALIB.APSvrsnn -
TAPE=SYS006 -
LIST=YES -
REPLACE=YES
/*
// MTC REW,SYS006
// ASSGN SYS006,UA
/&
* $$ EOJ ↵
```

- 5 Repeat the previous step for SAGLIB.APSvrs.
- 6 Modify the Adabas startup JCL, adding the UES environment section after the ADARUN parameters:

```

ADARUN .....
ADARUN ....
/*
TCPIP=NO
ENVIRONMENT_VARIABLES=/DDECSOJ/ADAvrs/ENVVARS.P
/*
/&
* $$ EOJ

```



**Note:** TCPIP=NO must always be specified in the APS parameter list, regardless of the ADARUN TCPIP parameter setting.

Reference the library where the libraries were restored in your Adabas startup procedure:

```

// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,SAG.ADABAS.LIB,2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt

```

And add the libraries to the LIBDEF chain:(be sure SAGLIB.APSvrsnn is referenced *before* SAGLIB.APSvrs):

```

// DLBL APSLIB,SAG.APS.LIB,2099/365,SD
// EXTENT SYS006,vvvvvv,1,0,ssss,tttt
// LIBDEF PHASE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs,...X
DDECSOJ.DDECSOJ, X
APSLIB.APSvrsnn,APSLIB.APSvrs)
// LIBDEF OBJ,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
DDECSOJ.APSvrsnn,DDECSOJ.APSvrs)
// LIBDEF SOURCE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs,... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
APSLIB.APSvrsnn,APSLIB.APSvrs)
// LIBDEF PHASE,CATALOG=SAGLIB.USRLIB

```

If you are running UES with TCP/IP support, set up the LIBDEF chain as follows (setup of the WCPvrs and WTCvrs libraries referenced in this chain is described in your Entire Net-Work documentation):

```

// DLBL APSLIB,SAG.APS.LIB,2099/365,SD
// EXTENT SYS006,vvvvvv,1,0,ssss,tttt
// LIBDEF PHASE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs,...X
SAGLIB.WCPvrs,SAGLIB.WTCvrs,... X
DDECSOJ.DDECSOJ, X
APSLIB.APSvrsnn,APSLIB.APSvrs)
// LIBDEF OBJ,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
SAGLIB.WCPvrs,SAGLIB.WTCvrs,... X

```

```
DDECSOJ.APSvrsnn,DDECSOJ.APSvrs)
// LIBDEF SOURCE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs,... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
APSLIB.APSvrsnn,APSLIB.APSvrs)
```

- 7 Modify the ENVVARS.P file, adding the following line in the APSvrs library:

```
* This member contains Environment Variables used by APS and
* APS-based applications.
*
ECSOBJDIR=FILE://DDECSOJ/DDECSOJ
```

- 8 Run the ADADEF utility setting UES=YES:

```
* $$ JOB JNM=ADADEF,CLASS=0,DISP=D,LDEST=(,xxxxxx)
* $$ LST CLASS=A,DISP=D
// JOB ADADEF EXECUTE THE ADABAS VERSION 7 UTILITY ***DEF***
// OPTION LOG,PARTDUMP
*
// EXEC PROC=ADALIB
*
// EXEC PROC=ADAFIL
*
// EXEC ADARUN,SIZE=ADARUN
*
ADARUN PROG=ADADEF,MODE=SINGLE,SVC=svc,DEVICE=dddd,DBID=nnnn
/*
ADADEF MODIFY UES=YES
/*
/&
* $$ EOJ
```

- 9 Start the database.

You should see the following message:

```
ENTIRE CONVERSION SERVICES INITIALIZED
```

# 7

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

---

- Installing TCP/IP Support for the Adabas Nucleus ..... 130

TCP/IP access to the database is performed using Entire Net-Work components. The Entire Net-Work sublibraries must be made accessible in the LIBDEF PHASE,SEARCH declaration.

## Installing TCP/IP Support for the Adabas Nucleus

---

The following Entire Net-Work LIBR sublibraries are distributed to provide TCP/IP support:

```
SAGLIB.WCPvrs
SAGLIB.WTCvrs
```

### ➤ To install these libraries:

- 1 Create a z/VSE sublibrary for the Entire Net-Work components.

```
* $$ JOB JNM=CRUWCP,CLASS=0,DISP=D,LDEST=(,xxxxxx)
* $$ LST CLASS=A,DISP=D
// JOB LIBRDEF
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL SAGLIB,'SAG.ADABAS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// EXEC LIBR
DEFINE L=WCPvrs      R=Y
DEFINE L=WTCvrs      R=Y
/*
/&
* $$ EOJ
```

Check that the SVC used by ADALNK in this example is correct. It can be set up by modifying the LNKGBLS source. After this, it may be necessary to modify the translation tables. For more information, read [LNKUES for Data Conversion](#), elsewhere in this section. These can be assembled and linked using the LNKLNK8 sample job.

- 2 Create a z/VSE sublibrary for APS.

```
* $$ JOB JNM=CRUAPS,CLASS=0,DISP=D,LDEST=(,xxxxxx)
* $$ LST CLASS=A,DISP=D
// JOB LIBRDEF
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,'SAG.APS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// EXEC LIBR
DEFINE L=APSvrsnn R=Y
DEFINE L=APSvrs    R=Y
/*
/&
* $$ EOJ ↵
```

- 3 Restore the UES code pages sublibrary to this file. Refer to the *Software AG Product Delivery Report* for the file positions on the distribution tape.

```
* $$ JOB JNM=RESTECS,DISP=D,CLASS=0,LDEST=(,xxxxxx)
* $$ LST DISP=D,CLASS=A
// JOB RESTECS
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL ADALIB,'SAG.ADABAS.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// ASSGN SYS006,cuu
// MTC REW,SYS006
// EXEC LIBR,PARM='MSHP'
RESTORE SUBLIB=SAGLIB.ADAvrCS:ADALIB.DDECSOJ -
TAPE=SYS006 -
LIST=YES -
REPLACE=YES
/*
// MTC REW,SYS006
// ASSGN SYS006,UA
/&
* $$ EOJ
```

- 4 Restore the APS sublibraries. Refer to the *Software AG Product Delivery Report* for the file positions on the distribution tape.

```
* $$ JOB JNM=RESTAPS,DISP=D,CLASS=0,LDEST=(,xxxxxx)
      * $$ LST DISP=D,CLASS=A
      // JOB RESTAPS
      // ASSGN SYS005,DISK,VOL=vvvvvv,SHR
      // DLBL APSLIB,'SAG.APS.LIB',2099/365,SD
      // EXTENT SYS005,vvvvvv,1,0,ssss,tttt
      // ASSGN SYS006,cuu
      // MTC REW,SYS006
      // EXEC LIBR,PARM='MSHP'
      RESTORE SUBLIB=SAGLIB.APSvrSnn:ADALIB.APSvrSnn -
      TAPE=SYS006 -
      LIST=YES -
      REPLACE=YES
      /*
      // MTC REW,SYS006
      // ASSGN SYS006,UA
      /&
      * $$ EOJ ↵
```

- 5 Repeat the previous step for SAGLIB.APSvrS, SAGLIB.WCPvrS and SAGLIB.WTCvrS.
- 6 Modify the Adabas startup JCL, adding the APS environment parameter list for UES after the ADARUN parameters:

```
ADARUN .....
ADARUN ....
/*
TCPIP=NO
ENVIRONMENT_VARIABLES=/DDECSOJ/ADAvrs/ENVVARS.P
/*
/&
* $$ EOJ
```



**Note:** TCPIP=NO must always be specified in the APS parameter list, regardless of the ADARUN TCPIP parameter setting.

Reference the library where the libraries were restored in your Adabas startup procedure:

```
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,SAG.ADABAS.LIB,2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
```

And add the libraries to the LIBDEF chain:(be sure SAGLIB.APSvrsnn is referenced *before* SAGLIB.APSvrs):

```
// DLBL APSLIB,SAG.APS.LIB,2099/365,SD
// EXTENT SYS006,vvvvvv,1,0,ssss,tttt
// LIBDEF PHASE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs,...X
DDECSOJ.DDECSOJ, X
APSLIB.APSvrsnn,APSLIB.APSvrs)
// LIBDEF OBJ,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
DDECSOJ.APSvrsnn,DDECSOJ.APSvrs)
// LIBDEF SOURCE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs,... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
APSLIB.APSvrsnn,APSLIB.APSvrs)
// LIBDEF PHASE,CATALOG=SAGLIB.USRLIB// LIBDEF ↵
PHASE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs,...X
SAGLIB.WCPvrs,SAGLIB.WTCvrs,... X
DDECSOJ.DDECSOJ,... X
APSLIB.APSvrsnn,APSLIB.APSvrs)
```

If you are running UES with TCP/IP support, set up the LIBDEF chain as follows (setup of the WCPvrs and WTCvrs libraries referenced in this chain is described in your Entire Net-Work documentation):



```
// DLBL APSLIB,SAG.APS.LIB,2099/365,SD
// EXTENT SYS006,vvvvvv,1,0,ssss,tttt
// LIBDEF PHASE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs,...X
SAGLIB.WCPvrs,SAGLIB.WTCvrs,... X
DDECSOJ.DDECSOJ, X
APSLIB.APSvrsnn,APSLIB.APSvrs)
// LIBDEF OBJ,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
SAGLIB.WCPvrs,SAGLIB.WTCvrs,... X
DDECSOJ.APSvrsnn,DDECSOJ.APSvrs)
// LIBDEF SOURCE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADAvrs,... X
SAGLIB.AOSvrs,SAGLIB.ADEvrs,SAGLIB.ACFvrs, X
APSLIB.APSvrsnn,APSLIB.APSvrs)
```

- 7 Modify the ENVVARS.P file, adding the following line in the APSvrs library:

```
* This member contains Environment Variables used by APS and
* APS-based applications.
*
ECSOJDIR=FILE://DDECSOJ/DDECSOJ
```

- 8 Run the ADADEF utility setting UES=YES:

```
* $$ JOB JNM=ADADEF,CLASS=0,DISP=D,LDEST=(,xxxxxx)
* $$ LST CLASS=A,DISP=D
// JOB ADADEF EXECUTE THE ADABAS VERSION 7 UTILITY ***DEF***
// OPTION LOG,PARTDUMP
*
// EXEC PROC=ADALIB
*
// EXEC PROC=ADAFIL
*
// EXEC ADARUN,SIZE=ADARUN
*
ADARUN PROG=ADADEF,MODE=SINGLE,SVC=svc,DEVICE=dddd,DBID=nnnn
/*
ADADEF MODIFY UES=YES
/*
/&
* $$ EOJ
```

- 9 Activate UES support in the Adabas nucleus. This is accomplished by ensuring that the following parameters have been set:
- The ADARUN TCPIP parameter has been set to "YES".
  - The ADARUN TCPURL parameter specifies an appropriate URL for the appropriate TCP/IP application programming interface (API).

Additional information is provided in *Installing UES Support for the Adabas Nucleus*, elsewhere in this section.

An example job for the nucleus can be found in the supplied library element ADANUCT(X).

10 Start the database.

You should see the following message:

```
ENTIRE CONVERSION SERVICES INITIALIZED
```

# 8

## Device and File Considerations

---

|   |     |
|---|-----|
| ■ Supported z/VSE Device Types .....      | 136 |
| ■ FBA Devices .....                       | 137 |
| ■ ECKD Devices .....                      | 138 |
| ■ Adding New Devices .....                | 138 |
| ■ User ZAPs to Change Logical Units ..... | 142 |

This section provides information regarding device- and system file-related topics, such as:

- installing on fixed-block addressing (FBA) devices;
- defining new devices; and
- changing defaults for sequential files.

## Supported z/VSE Device Types

The standard characteristics of the device types supported by Adabas are summarized in the following table. The Adabas block sizes and RABNs per track are provided for each component for each device type.

| Device | Trks/Cyl | ASSO    | DATA     | WORK     | PLOG/RLOG | CLOG     | TEMP/SORT/DSIM | Notes |
|--------|----------|---------|----------|----------|-----------|----------|----------------|-------|
| 1512   | 7        | 1536:37 | 18944:37 | 18944:37 | 18944:37  | 18944:37 | 18944:37       |       |
| 3375   | 12       | 2016:15 | 4092:8   | 4096:8   | 4096:8    | 4096:8   | 8608:4         |       |
| 3380   | 15       | 2004:19 | 4820:9   | 5492:8   | 5492:8    | 4820:9   | 7476:6         | 2     |
| 3390   | 15       | 2544:18 | 5064:10  | 5724:9   | 5724:9    | 5064:10  | 8904:6         | 2     |
| 5121   | 15       | 2048:16 | 4096:8   | 4096:8   | 4096:8    | 4096:8   | 4096:8         |       |
| 5122   | 15       | 4096:8  | 8192:4   | 8192:4   | 8192:4    | 8192:4   | 8192:4         |       |
| 5123   | 15       | 4096:8  | 16384:2  | 16384:2  | 16384:2   | 16384:2  | 16384:2        |       |
| 8345   | 15       | 4092:10 | 22780:2  | 22920:2  | 22920:2   | 22920:2  | 22920:2        |       |
| 8380   | 15       | 3476:12 | 6356:7   | 9076:5   | 9076:5    | 9076:5   | 9076:5         | 1     |
| 8381   | 15       | 3476:12 | 9076:5   | 11476:4  | 11476:4   | 9076:5   | 9076:5         | 1     |
| 8385   | 15       | 4092:10 | 23292:2  | 23468:2  | 23468:2   | 23468:2  | 23468:2        | 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     |
| 9345   | 15       | 4092:10 | 7164:6   | 11148:4  | 11148:4   | 22920:2  | 22920:2        | 2     |



### Notes:

1. The 8350, 838<sub>n</sub>, and 839<sub>n</sub> are pseudo-device types physically contained on a 3350, 3380, and 3390 device, respectively, but for which some or all of the standard block sizes are larger.
2. The IBM RAMAC 9394 emulates devices 3390 Model 3, 3380 Model K, or 9345 Model 2.

## FBA Devices

All device definitions for Adabas control statements for FBA disks should specify one of the following devices types:

- FBA SCSI devices: Specify a device type of 1512.
- Virtual FBA devices: Specify device types of 5121, 5122, or 5123.



**Note:** Virtual FBA devices are not permanent and are, therefore, only suitable for holding temporary or work data sets.

Choose a device type based on the block sizes given in the following tables:

### SCSI Device Types:

| Dev Type | Asso blksz | Data blksz | Work blksz | Temp blksz | Sort blksz | PLOG blksz | CLOG blksz |
|----------|------------|------------|------------|------------|------------|------------|------------|
| 1512     | 1536       | 18944      | 18944      | 18944      | 18944      | 18944      | 18944      |

### Virtual FBA Device Types:

| Dev Type | Asso blksz | Data blksz | Work blksz | Temp blksz | Sort blksz | PLOG blksz | CLOG blksz |
|----------|------------|------------|------------|------------|------------|------------|------------|
| 5121     | 2048       | 4096       | 4096       | 4096       | 4096       | 4096       | 4096       |
| 5122     | 4096       | 8192       | 8192       | 8192       | 8192       | 8192       | 8192       |
| 5123     | 4096       | 16384      | 16384      | 16384      | 16384      | 16384      | 16384      |

The pseudo-cylinder for each of these devices has a different number of blocks as described below:

- 1512 cylinder = FBA blocks/777
- 5121 cylinder = FBA blocks/960
- 5122 cylinder = FBA blocks/960
- 5123 cylinder = FBA blocks/960

The size definitions for FBA devices on Adabas control statements can specify the number of pseudo-cylinders or the number of Adabas blocks (RABNs).

Make sure that the starting block and the number of FBA blocks on the z/VSE EXTENT statement are on an FBA pseudo-cylinder boundary, which is based on the device as specified above for each Adabas file comprising the database:

- An SCSI pseudo-cylinder (device type 1512) comprises 777 elements of 512 bytes each, or 388K per pseudo-cylinder. For example, an EXTENT entry for a ten cylinder SCSI device might consist of:

```
// EXTENT SYS123,,,,,777,7770
```

- A virtual FBA pseudo-cylinder comprises 960 elements of 512 bytes each, or 480K per pseudo-cylinder. For example, an EXTENT entry for a ten cylinder virtual FBA device might consist of:

```
// EXTENT SYS123,,,,,512,5120
```

## ECKD Devices

---

Adabas supports ECKD DASD devices such as the IBM 3390 with the 3990 controller and ESCON channels.

During an open operation, ADAIOR determines which DASD device types are being used for the ASSO, DATA, WORK, SORT, and TEMP data sets. At that time, Adabas issues an informational message for each Adabas database component, where *type* is the component:

```
ADA164 ... FILE DDtype HAS BEEN OPENED IN ckd/eckd MODE - RABN SIZE rabn-size
```



**Note:** Software AG strongly recommends that you avoid mixing ECKD and CKD extents within a file, because the file will be opened only in CKD mode. Mixing extents could degrade performance when file I/O operations are performed.

## Adding New Devices

---

Support for new device types that include user-defined block sizes can be implemented in ADAIOR by modifying one of the table of device-constant entries (TDCEs) reserved for this purpose.

A TDCE is X'40' bytes long and the first free TDCE can be identified by X'0000' in its first two bytes (TDCDT).

For Adabas Version 8, TDCE entries are in the ADAIOS CSECT TDCON, which corresponds to ESDID 1EC in object module IOSVSE.OBJ. The first TDCE entry is at offset X'19398' into IOSVSE.OBJ; the first free TDCE entry is at offset X'19898'.

This information is valuable when adding an additional TDCE entry, and when zapping the object module and relinking ADAIOS under z/VSE.

The z/VSE MSHP control statements to add a TDCE entry at the first free entry thus take the form:

```
// EXEC MSHP
CORRECT 9001-ADA-00-vrs :AD99998
AFFECTS MODULE=IOSVSE,ESDID=1EC
ALTER 19898 0000 : nnnn
ALTER 1989A 0000 : nnnn
.
. (etc.)
.
INVOLVES LINK=LNKIOS
/*
```

- [Information to be Zapped into the First Free ADAIOR TDCE](#)
- [General Rules for Defining Device Block Sizes](#)
- [Using 3480/3490 Tape Cartridge Compression \(IDRC\)](#)

### Information to be Zapped into the First Free ADAIOR TDCE

The information in the following tables must be zapped into the first free TDCE. The rules described in the section [General Rules for Defining Device Block Sizes](#) must be followed when changing the TDCE.

| Label   | Offset | Contents   |
|---------|--------|--|
| TDCDT   | 00     | Device type in unsigned decimal (X'3385'), must be numeric, and unique among all TDCEs.  |
| TDCKSN  | 02     | Constant set number: must be uniquely chosen from the values X'2B' or X'2E'.   |
| TDCF    | 03     | The flag bit must be set—TDCFCKD (X'40') for CKD devices, TDCFECKD (X'60') for ECKD devices or TDCFECKD (X'61') for ECKD, not user defined devices.                      |
| TDCDT1  | 04     | (see note)   |
| TDCDT2  | 05     | (see note)   |
| TDCDT3  | 06     | (see note)   |
| TDCDT4  | 07     | (see note)   |
| TDCMSBS | 08     | Refer to the TDCMSBS default table in <i>Maximum Sequential Block Size</i> in the Adabas z/OS installation instructions for more system- and device-related information. |
| TDCTPC  | 0A     | Number of tracks per cylinder.   |
| TDCCIPT | 0C     | Number of FBA blocks or PAM pages per track (if TDCFFBA is set).   |
| TDCBPCI | 0E     | Number of bytes per FBA block or PAM page (2048 if TDCFFBA is set).  |
| TDCABPT | 10     | Number of Associator blocks per track.   |
| TDCABS  | 12     | Associator block size.   |
| TDCACPB | 14     | Number of FBA blocks or PAM pages per Associator block (if TDCFFBA is set).  |
| TDCDBPT | 16     | Number of Data Storage blocks per track.   |
| TDCDBS  | 18     | Data Storage block size.   |
| TDCDCPB | 1A     | Number of FBA blocks or PAM pages per Data Storage block (if TDCFFBA is set).  |
| TDCWBPT | 1C     | Number of Work blocks per track.   |

| Label    | Offset | Contents  |
|----------|--------|---|
| TDCWBS   | 1E     | Work block size.  |
| TDCWCPB  | 20     | Number of FBA blocks or PAM pages per Work block (if TDCFFBA is set).         |
| TDCTSBPT | 22     | Number of TEMP or SORT blocks per track                                       |
| TDCTSBS  | 24     | TEMP or SORT block size.  |
| TDCTSCP  | 26     | Number of FBA blocks or PAM pages per TEMP or SORT block (if TDCFFBA is set). |
| TDCPBPT  | 28     | Number of PLOG blocks per track.  |
| TDCPBS   | 2A     | PLOG block size.  |
| TDCPCPB  | 2C     | Number of FBA blocks or PAM pages per PLOG block (if TDCFFBA is set).         |
| TDCCBPT  | 2E     | Number of CLOG blocks per track.  |
| TDCCBS   | 30     | CLOG block size.  |
| TDCCCPB  | 32     | Number of FBA blocks or PAM pages per CLOG block (if TDCFFBA is set).         |



**Note:** One or more z/VSE codes for identifying the device type: PUB device type from PUBDEVTY (refer to the IBM MAPDEVTY macro).

### General Rules for Defining Device Block Sizes

The following general rules must be followed when defining Adabas device block sizes:

- All block sizes must be multiples of 4.
- A single block cannot be split between tracks (that is, the block size must be less than or equal to the track size).

### Block Rules for ASSO/DATA

The following rules are applicable for Associator and Data Storage:

- Associator block size must be greater than one-fourth the size of the largest FDT, and should be large enough to accept definitions in the various administrative blocks (RABN 1 - 30) and in the FCB;
- The block sizes for Associator and Data Storage should be a multiple of 256, less four bytes (for example, 1020) to save Adabas buffer pool space.
- The Associator and Data Storage block sizes must be at least 32 less than the sequential block size.
- Data Storage block size must be greater than: (maximum compressed record length + 10 + padding bytes).

### Block Rule for WORK

The following rule is applicable for Work::



- The Work block size must be greater than either (maximum compressed record length + 110) or (Associator block size + 110), whichever is greater.

### Block Rules for TEMP/SORT

The following rules are applicable for TEMP and SORT:

- Block sizes for TEMP and SORT must be greater than the block sizes for Data Storage.
- If ADAM direct addressing is used:

```
size > (maximum compressed record length + ADAM record length + 24);
size > 277 (maximum descriptor length + 24)
```

TEMP and SORT are generally read and written sequentially; therefore, the larger the TEMP/SORT block size, the better.

### Block Rule for PLOG or SIBA

The following rules are applicable for PLOG and SIBA:

- The PLOG or SIBA block size must be greater than either (maximum compressed record length + 110) or (Associator block size + 110), whichever is greater.
- It is also recommended that PLOG/SIBA be defined larger than the largest Data Storage block size. This avoids increased I/O caused by splitting Data Storage blocks during online ADASAV operations.

The block size (BLKSIZE) of a sequential file is determined as follows:

```
if PTF(JCL) then BLKSIZE is taken from file assignment statement or label;
if PTTMBS > 0 then BLKSIZE = PTTMBS;
if PTTMBS = 0 then
  if tape then BLKSIZE = 32760;
  else BLKSIZE = TDCMSBS;
else if BLKSIZE in file assignment statement or label then use it;
if PTF(OUT) then
  if QBLKSIZE > 0 then BLKSIZE = QBLKSIZE;
  if tape then BLKSIZE = 32760;
  else BLKSIZE = TDCMSBS;
else error.
```



**Note:** QBLKSIZE is an ADARUN parameter.

## Using 3480/3490 Tape Cartridge Compression (IDRC)

The use of hardware compression (IDRC) is not recommended for protection log files. The ADARES BACKOUT function will run much longer when processing compressed data. Also, the BACKOUT function is not supported for compressed data.

## User ZAPs to Change Logical Units

The specified zaps should be added to the module IOSVSE / phase ADAIOS, not to the specified utility.

PTT entries are in the ADAIOS CSECT I\_PTT. The first PTT entry is at offset 0 into CSECT I\_PTT.

When zapping the object module and relinking ADAIOS, note that the ADAIOS CSECT I\_PTT corresponds to ESDID 000D in object module IOSVSE.OBJ. The offset of CSECT I\_PTT into IOSVSE.OBJ is X'1000'.

| Utility | File           | Default SYS Number | PTT Offset | VER  | REP  |
|---------|----------------|--------------------|------------|------|------|
| ADACDC  | SIIN           | SYS010             | 6B8        | 1A0A | 1Axx |
| ADACMP  | AUSBA          | SYS012             | 008        | 820C | 82xx |
|         | AUSB1          | SYS021             | 018        | 8215 | 82xx |
|         | AUSB2          | SYS022             | 028        | 8216 | 82xx |
|         | AUSB3          | SYS023             | 038        | 8217 | 82xx |
|         | AUSB4          | SYS024             | 048        | 8218 | 82xx |
|         | AUSB5          | SYS025             | 058        | 8219 | 82xx |
|         | AUSB6          | SYS026             | 068        | 821A | 82xx |
|         | AUSB7          | SYS027             | 078        | 821B | 82xx |
|         | AUSB8          | SYS028             | 088        | 821C | 82xx |
|         | AUSB9          | SYS029             | 098        | 821D | 82xx |
|         | EBAND          | SYS010             | 0A8        | 180A | 18xx |
|         | FEHL           | SYS014             | 0B8        | 820E | 82xx |
| ADACNV  | FILEA (OUTPUT) | SYS010             | 698        | 820A | 82xx |
|         | FILEA (INPUT)  | SYS010             | 6A8        | 120A | 12xx |
| ADALOD  | FILEA (OUTPUT) | SYS012             | 0D8        | 820C | 82xx |
|         | FILEA (INPUT)  | SYS012             | 0E8        | 020C | 02xx |
|         | EBAND          | SYS010             | 0F8        | 1A0A | 1Axx |
|         | ISN            | SYS016             | 108        | 1A10 | 1Axx |
|         | LOB (OUTPUT)   | SYS017             | 118        | 8211 | 82xx |
|         | LOB (INPUT)    | SYS017             | 128        | 1A11 | 1Axx |

| Utility | File           | Default SYS Number | PTT Offset | VER  | REP  |
|---------|----------------|--------------------|------------|------|------|
|         | OLD            | SYS014             | 138        | 820E | 82xx |
| ADAMER  | EBAND          | SYS010             | 148        | 1A0A | 1Axx |
| ADANUC  | LOG            | SYS012             | 158        | 820C | 82xx |
|         | SIBA           | SYS014             | 168        | C20E | C2xx |
| ADAORD  | FILEA (OUTPUT) | SYS010             | 178        | 820A | 82xx |
|         | FILEA (INPUT)  | SYS010             | 188        | 120A | 12xx |
| ADAPLP  | PLOG           | SYS014             | 198        | 1A0E | 1Axx |
| ADARAI  | OUT            | SYS010             | 6C8        | 800A | 80xx |
| ADAREP  | SAVE           | SYS010             | 1A8        | 1A0A | 1Axx |
|         | PLOG           | SYS011             | 1B8        | 1A0B | 1Axx |
| ADARES  | SIIN           | SYS020             | 1C8        | 1A14 | 1Axx |
|         | BACK           | SYS020             | 1D8        | 2C14 | 2Cxx |
|         | SIAUS1         | SYS021             | 1E8        | 8215 | 82xx |
|         | SIAUS2         | SYS022             | 1F8        | 8216 | 82xx |
| ADASAV  | SAVE1          | SYS011             | 208        | 820B | 82xx |
|         | SAVE2          | SYS012             | 218        | 820C | 82xx |
|         | SAVE3          | SYS013             | 228        | 820D | 82xx |
|         | SAVE4          | SYS014             | 238        | 820E | 82xx |
|         | SAVE5          | SYS015             | 248        | 820F | 82xx |
|         | SAVE6          | SYS016             | 258        | 8210 | 82xx |
|         | SAVE7          | SYS017             | 268        | 8211 | 82xx |
|         | SAVE8          | SYS018             | 278        | 8212 | 82xx |
|         | DUAL1          | SYS021             | 288        | 8215 | 82xx |
|         | DUAL2          | SYS022             | 298        | 8216 | 82xx |
|         | DUAL3          | SYS023             | 2A8        | 8217 | 82xx |
|         | DUAL4          | SYS024             | 2B8        | 8218 | 82xx |
|         | DUAL5          | SYS025             | 2C8        | 8219 | 82xx |
|         | DUAL6          | SYS026             | 2D8        | 821A | 82xx |
|         | DUAL7          | SYS027             | 2E8        | 821B | 82xx |
|         | DUAL8          | SYS028             | 2F8        | 821C | 82xx |
|         | REST1          | SYS011             | 308        | 1A0B | 1Axx |
|         | REST2          | SYS012             | 318        | 120C | 12xx |
|         | REST3          | SYS013             | 328        | 120D | 12xx |
|         | REST4          | SYS014             | 338        | 120E | 12xx |
|         | REST5          | SYS015             | 348        | 120F | 12xx |
|         | REST6          | SYS016             | 358        | 1210 | 12xx |

| Utility | File   | Default SYS Number | PTT Offset | VER  | REP   |
|---------|--------|--------------------|------------|------|-------|
|         | REST7  | SYS017             | 368        | 1211 | 12xx  |
|         | REST8  | SYS018             | 378        | 1212 | 12xx  |
|         | FULL   | SYS030             | 388        | 1A1E | 1Axx  |
|         | DEL1   | SYS031             | 398        | 1A1F | 1Axx  |
|         | DEL2   | SYS032             | 3A8        | 1A20 | 1Axx  |
|         | DEL3   | SYS033             | 3B8        | 1A21 | 1Axx  |
|         | DEL4   | SYS034             | 3C8        | 1A22 | 1Axx  |
|         | DEL5   | SYS035             | 3D8        | 1A23 | 1Axx  |
|         | DEL6   | SYS036             | 3E8        | 1A24 | 1Axx  |
|         | DEL7   | SYS037             | 3F8        | 1A25 | 1Axx  |
|         | DEL8   | SYS038             | 408        | 1A26 | 1Axx  |
|         | PLOG   | SYS010             | 418        | 1A0A | 1Axx  |
| ADASEL  | EXPA1  | SYS011             | 428        | 820B | 82xx  |
|         | EXPA2  | SYS012             | 438        | 820C | 82xx  |
|         | EXPA3  | SYS013             | 448        | 820D | 82xx  |
|         | EXPA4  | SYS014             | 458        | 820E | 82xx  |
|         | EXPA5  | SYS015             | 468        | 820F | 82xx  |
|         | EXPA6  | SYS016             | 478        | 8210 | 82xx  |
|         | EXPA7  | SYS017             | 488        | 8211 | 82xx  |
|         | EXPA8  | SYS018             | 498        | 8212 | 82xx  |
|         | EXPA9  | SYS019             | 4A8        | 8213 | 82xx  |
|         | EXPA10 | SYS020             | 4B8        | 8214 | 82xx  |
|         | EXPA11 | SYS021             | 4C8        | 8215 | 82xx  |
|         | EXPA12 | SYS022             | 4D8        | 8216 | 82xx  |
|         | EXPA13 | SYS023             | 4E8        | 8217 | 82xx  |
|         | EXPA14 | SYS024             | 4F8        | 8218 | 82xx  |
|         | EXPA15 | SYS025             | 508        | 8219 | 82xx  |
|         | EXPA16 | SYS026             | 518        | 821A | 82xx  |
|         | EXPA17 | SYS027             | 528        | 821B | 82xx  |
|         | EXPA18 | SYS028             | 538        | 821C | 82xx  |
|         | EXPA19 | SYS029             | 548        | 821D | 82xx  |
|         | EXPA20 | SYS030             | 558        | 821E | 82xx  |
|         | SIIN   | SYS010             | 568        | 1A0A | 1Axx  |
| ADATRA  | TRA    | SYS019             | 578        | 820A | 82xx  |
| ADAULD  | OUT1   | SYS010             | 588        | 820A | 820xx |
|         | OUT2   | SYS011             | 598        | 820B | 820xx |

| Utility | File | Default SYS Number | PTT Offset | VER  | REP   |
|---------|------|--------------------|------------|------|-------|
|         | ISN  | SYS012             | 5A8        | 820C | 820xx |
|         | SAVE | SYS013             | 5B8        | 1A0D | 1Axx  |
|         | PLOG | SYS014             | 5C8        | 1A0E | 1Axx  |
|         | FULL | SYS030             | 5D8        | 1A1E | 1Axx  |
|         | DEL1 | SYS031             | 5E8        | 1A1F | 1Axx  |
|         | DEL2 | SYS032             | 5F8        | 1A20 | 1Axx  |
|         | DEL3 | SYS033             | 608        | 1A21 | 1Axx  |
|         | DEL4 | SYS034             | 618        | 1A22 | 1Axx  |
|         | DEL5 | SYS035             | 628        | 1A23 | 1Axx  |
|         | DEL6 | SYS036             | 638        | 1A24 | 1Axx  |
|         | DEL7 | SYS037             | 648        | 1A25 | 1Axx  |
|         | DEL8 | SYS038             | 658        | 1A26 | 1Axx  |
| ADAAVAL | FEHL | SYS014             | 668        | 820E | 820xx |



## 9 Installing The AOS Demo Version

---

|   |     |
|---|-----|
| ■ AOS Demo Installation Procedure .....       | 148 |
| ■ Installing AOS with Natural Security .....  | 149 |
| ■ Setting the AOS Demo Version Defaults ..... | 150 |

This section describes how to install the Adabas Online System (AOS) demo version. To install AOS on systems that use Software AG'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.

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.



**Note:** To install the full version of Adabas Online System (AOS), read the *Adabas Online System* documentation.

## 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 medium 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.

- 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 installation medium.

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

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

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.

➤ **Use the following procedure if Natural Security is installed in your environment.**

- 1 Define at least the library `SYSAOS` to Natural Security

Software AG recommends you define this library and any others you may define as protected.

- 2 Specify the startup program for `SYSAOS` as `MENU`

*Do not* specify a startup program name for the other libraries.

## 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   |
| CMD-INT     | Natural (N) or AOS ( A )  | A       | Pass-through control for invalid AOS commands:<br>"N" passes invalid commands to Natural;<br>"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.   |
| 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.   |

| Parameter   | Valid Values       | Default            | Description  |
|-------------|--------------------|--------------------|--|
| 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?   |
| 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:<br>"B" = binary TID display;<br>"A" = alpha TID display;<br>"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)
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
```

# 10

## Installing The Recovery Aid (ADARAI)

---

|                                       |     |
|---------------------------------------|-----|
| ■ ADARAI Installation Overview .....  | 154 |
| ■ ADARAI Installation Procedure ..... | 154 |

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 Allocate the recovery logs

Define and format the RLOGR1 file.

Use the ADAFRM RLOGFRM function to format the RLOGs.

#### 2 Add data definition statements

Add an RLOGR1 DLBL statement to the nucleus job stream and to any utilities that update or save the database and thus write to the RLOG files.

Whenever these utilities are executed while ADARAI is active in the database (that is, after the PREPARE function has been executed), the RLOGR1 DLBL 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.





# 11

## Adabas Dump Formatting Tool (ADAFDP)

---

|                         |     |
|-------------------------|-----|
| ■ ADAFDP Function ..... | 158 |
| ■ ADAFDP Output .....   | 158 |

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   |
|----------------------------------|---|
| ADAH51 / ADAH52                  | 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 <a href="#">Pool Abbreviations</a> 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 <a href="#">User Threads</a> 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 <a href="#">Command Information</a> 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 <a href="#">RABN Information</a> 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               | Software AG 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               | Software AG 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            | <p>Indicates the current status of the thread:</p> <ul style="list-style-type: none"> <li>■ *Active*: the currently active thread</li> <li>■ In Use: thread has been assigned work</li> <li>■ Waiting For I/O: waiting for a block not in buffer pool</li> <li>■ Waiting For RABN: waiting for a RABN already in use</li> <li>■ Waiting For Work-2 Area Block: similar to waiting for I/O</li> <li>■ Waiting Workpool Space: provides number of bytes in decimal</li> <li>■ Ready To Run: waiting to be selected for execution</li> </ul> |
| 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<br>RL: record buffer length<br>SL: search buffer length<br>VL: value buffer length<br>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           | <p>Command queue flag bytes:</p> <ul style="list-style-type: none"> <li>■ First Byte: General Purpose Flag <ul style="list-style-type: none"> <li>■ X'80': User buffers in service partition, region, address space</li> <li>■ X'40': ET command waiting for 12 call</li> <li>■ X'20': Waiting for 16 call</li> <li>■ X'10': 16 call required</li> <li>■ X'08': Attached buffer</li> <li>■ X'04': Attached buffer required</li> <li>■ X'02': X-memory lock held (z/OS only)</li> </ul> </li> <li>■ Second Byte: Selection Flag <ul style="list-style-type: none"> <li>■ X'80': In process</li> <li>■ X'40': Ready to be selected</li> <li>■ X'20': Search for UQE done</li> <li>■ X'10': UQE found</li> <li>■ X'08': Not selectable during BSS=x'80' status</li> <li>■ X'04': Not selectable during ET-SYNC</li> <li>■ X'02': Waiting for space</li> <li>■ X'01': Waiting for ISN in HQ</li> </ul> </li> </ul> |
| 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: <ul style="list-style-type: none"> <li>■ AKZ X'40': Active indicator</li> <li>■ UKZ X'20': Update indicator</li> <li>■ RKZ X'10': Read indicator</li> <li>■ XKZ X'04': Access is waiting for block</li> <li>■ YKZ X'02': Update is waiting for block</li> <li>■ SKZ X'01': Write indicator</li> </ul> |
| File        | File number that owns this block   |
| Address     | Address location of block in storage.  |





# 12

## Maintaining A Separate Test Environment

---

This section describes a method to set up a temporary test copy of phases updated by a program fix. The method described is intended as an example. Its relevance depends on the installation standards you use for library maintenance.

The example scenario uses MSHP in a single z/VSE machine to control both the standard production Adabas library and an additional testing library or sublibrary used to validate recently applied program fixes.

After restoring the standard Adabas library and defining it to MSHP, an additional test library or sublibrary can be defined.

Object modules can then be copied from the standard library as required, and controlled with MSHP using a different z/VSE system history file. Using the same component ID as for the standard environment (9001-ADA-00-*vrs*) ensures that the ZAP source remains common to both environments.

The test version of a phase is then invoked by placing the test library or sublibrary at the head of the LIBDEF PHASE search chain.

The setup jobs required to implement this environment are described in detail below. Note that the first three steps form part of the standard installation process.

### » to setup the separate test environment:

- 1 Define standard Adabas library.

For a sample job, see the section [Installing the Adabas Release Tape](#).

- 2 Restore standard Adabas library.

For a sample job, see the section [Installing the Adabas Release Tape](#).

- 3 Define standard Adabas to MSHP.



**Note:** This job uses the history file identified by the IJSYSHF label in the z/VSE standard label area.

```
// EXEC MSHP
ARCHIVE ADAvrs
COMPRISES 9001-ADA-00
RESOLVES 'SOFTWARE AG - ADABAS.ADAvrs'
ARCHIVE 9001-ADA-00-vrs
RESIDENCE PRODUCT=ADAvrs -
PRODUCTION=SAGLIB.adannn -
GENERATION=SAGLIB.adannn
/*
```

where *vrs* is the Adabas version, revision, and system maintenance (SM) level and *adannn* is the sublibrary name for standard Adabas.

- 4 Create test sublibrary and copy object modules to it.

```
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// EXEC LIBR
DEFINE SUBLIB=SAGLIB.adatst
CONNECT SAGLIB.adannn:SAGLIB.adatst
COPY *.OBJ LIST=Y REPLACE=Y
/*
```

where *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, *adannn* is the sublibrary name for standard Adabas, and *adatst* is the sublibrary name for testing Adabas.

- 5 Create additional system history file for test environment and define test Adabas to it.

```
// ASSGN SYS020,DISK,VOL=volhis,SHR
// EXEC MSHP
CREATE HISTORY SYSTEM
DEFINE HISTORY SYSTEM EXTENT=start:numtrks -
UNIT=SYS020 -
ID='sag.test.system.history.file'
ARCHIVE ADAvrs
COMPRISES 9001-ADA-00
RESOLVES 'SOFTWARE AG - ADABAS Vvrs'
ARCHIVE 9001-ADA-00-vrs
RESIDENCE PRODUCT=ADAvrs -
PRODUCTION=SAGLIB.adatst -
GENERATION=SAGLIB.adatst
/*
```

where *volhis* is the volume on which test system history file resides, *start* is the start of extent on which test system history file resides, *numtrks* is the length of extent on which test system history file resides, *sag.test.system.history.file* is the physical name of test system history file, *vrs* is the Adabas *version*, and *adatst* is the sublibrary name for testing Adabas.

## 6 Apply zap to test environment.

```
// DLBL IJSYSHF,'sag.test.system.history.file'
// EXTENT SYS020,,,start,numtrks
// ASSGN SYS020,DISK,VOL=volhis,SHR
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// EXEC MSHP
CORRECT 9001-ADA-00-vrs : ADnnnnn
AFFECTS MODULE=modname
ALTER offset hexold : hexnew
INVOLVES LINK=lnkname
/*
```

where *sag.test.system.history.file* is the physical name of test system history file, *start* is the start of extent on which test system history file resides, *numtrks* is the length of extent on which test system history file resides, *volhis* is the volume on which test system history file resides, *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, *vrs* is the Adabas *version*, *nnnnn* is the Adabas fix number, *modname* is the Adabas object module to be zapped and then relinked, *offset* is the hexadecimal offset to the beginning of the zap, *hexold* is the verify data for the zap, *hexnew* is the replace data for the zap, and *lnkname* is the link book for the phase affected.

## 7 Invoke updated test phase.

```
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// LIBDEF PHASE,SEARCH=(SAGLIB.adatst,SAGLIB.adannn,...)
...
```

where *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, *adatst* is the sublibrary name for testing Adabas, and *adannn* is the sublibrary name for standard Adabas.

## 8 Apply zap to standard environment.



**Note:** This job uses the history file identified by the IJSYSHF label in the z/VSE standard label area.

```
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// EXEC MSHP
CORRECT 9001-ADA-00-vrs : ADnnnnn
AFFECTS MODULE=modname
ALTER offset hexold : hexnew
INVOLVES LINK=lnkname
/*
```

where *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, *vrs* is the Adabas *version*, *nnnnn* is the Adabas fix number, *modname* is the Adabas object module to be zapped and then relinked, *offset* is the hexadecimal offset to the beginning of the zap, *hexold* is the verify data for the zap, *hexnew* is the replace data for the zap, and *lnkname* is the link book for the phase affected.

### 9 Invoke standard phase.

```
// DLBL SAGLIB,'adabas.adannn.library'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=volser,SHR
// LIBDEF PHASE,SEARCH=(SAGLIB.adannn,...)
...
```

where *adabas.adannn.library* is the physical name of the standard Adabas library, *volser* is the volume on which library resides, and *adannn* is the sublibrary name for standard Adabas.

# 13

## Translation Tables

---

|   |     |
|---|-----|
| ■ Adabas EBCDIC to ASCII and ASCII to EBCDIC .....          | 170 |
| ■ Entire Net-Work EBCDIC to ASCII and ASCII to EBCDIC ..... | 171 |

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'203F3F3F3F3F3F3F3F3F2E3C282B3F' 4.
c DC x'263F3F3F3F3F3F3F3F3F21242A293B5E' 5.
c DC x'2D2F3F3F3F3F3F3F3F3F7C2C255F3E3F' 6.
c DC x'3F3F3F3F3F3F3F3F3F603A2340273D22' 7.
c DC x'3F6162636465666768693F3F3F3F3F' 8.
c DC x'3F6A6B6C6D6E6F7071723F3F3F3F3F' 9.
c DC x'3F7E737475767778797A3F3F3F5B3F3F' A.
c DC x'3F3F3F3F3F3F3F3F3F3F3F3F5D3F3F' B.
c DC x'7B4142434445464748493F3F3F3F3F' C.
c DC x'7D4A4B4C4D4E4F5051523F3F3F3F3F' D.
c DC x'5C3F535455565758595A3F3F3F3F3F' E.
c DC x'303132333435363738393F3F3F3F3F' 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'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' 8.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' 9.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' A.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' B.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' C.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' D.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' E.
c DC x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F' 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'000000000000000000000000000000' 2.
DC X'000000000000000000000000000000' 3.
DC X'20000000000000000000000005B2E3C282B5D' 4.
DC X'26000000000000000000000021242A293B5E' 5.
DC X'2D2F00000000000000000007C2C255F3E3F' 6.
DC X'00000000000000000000000603A2340273D22' 7.
DC X'00616263646566676869000000000000' 8.
DC X'006A6B6C6D6E6F707172000000000000' 9.
DC X'007E737475767778797A000005B000000' A.
DC X'000000000000000000000000000005D0000' B.
DC X'7B414243444546474849000000000000' C.
DC X'7D4A4B4C4D4E4F505152000000000000' D.
DC X'5C7E535455565758595A000000000000' E.
DC X'303132333435363738397C00000000FF' 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'000000000000000000000000000000' 8.
DC X'000000000000000000000000000000' 9.
DC X'000000000000000000000000000000' A.
DC X'000000000000000000000000000000' B.
DC X'000000000000000000000000000000' C.
DC X'000000000000000000000000000000' D.
DC X'000000000000000000000000000000' E.
DC X'0000000000000000000000000000FF' F.
* .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
END

```





# Index

---

## A

- ACINAMES module, 65, 75
- ACIOPT table, 65
- Adabas
  - CICS execution unit, 64
  - installation for z/VSE, 5
- Adabas Bridge for DL/I support, 109
- Adabas Bridge for VSAM support, 109
- Adabas CICS task-related user exit (TRUE)
  - module name, 119
- Adabas Online System (AOS)
  - AOSEX1 program parameters, 150
  - modify default parameter values, 150
  - setting defaults, 150
- Adabas Review hub ID support, 116
- Adabas Review release, 117
- Adabas Review support, 116
- Adabas security interface parameter, 118
- Adabas SVC number parameter, 118
- Adabas Transaction Manager and Adabas Fastpath exit support, 110
- ADACICS module, 65, 76
- ADACICT module, 75
- ADADCI module, 65
- ADATCP support
  - enabling in z/VSE, 129
- ADL parameter, 109
- ADMIN-LEVEL parameter, 150
- AOS-END-MSG parameter, 150
- AOS-LOGO parameter, 150
- AOSEX1 user exit, 150
  - setting defaults, 150
- AVB parameter, 109

## B

- BATCH-ERROR parameter, 150
- BLS-CYL parameter, 150
- BS2000 IDT common memory pool name, 112
- BS2000 memory pool user bound setting, 112

## C

- CICS application stub, 65, 76
- CICS command-level link routine name, 111
- CICS execution unit, 64
- CICS installation options table, ACIOPT, 65, 74

- CICS multiple region option, 113
- CICS names module, ACINAMES, 65, 75
- CICS purge transaction parameter, 115
- CICS Resource Manager Interface parameter, 117
- CICS user ID creation method, 113
- CICS XWAIT setting, 122
- CITSNM parameter, 110
- CMD-INT parameter, 150
- CNAME parameter, 68
- command retry exit name, 117
- COR parameter, 110
- CPEXLIST parameter, 150
- CSECT or DSECT generation, 111

## D

- DBID parameter, 105
- DBID/SVC routing table, 110-111
  - source code, 103
- DBID2 parameter, 106
- DBSVCTN parameter, 110
- default target database ID, 112
- defaults, 150
- DSECT data prefix parameter, 115
- DYNDBSVC parameter, 111

## E

- ENTPT parameter, 77, 111
- ENTRY=FINAL statement, 70
- ENTRY=GLOBAL statement, 68
- ENTRY=GROUP statement, 69
- EX1-A1 parameter, 150
- EX1-N3 parameter, 150
- EXF-UTI parameter, 150

## G

- GBLNAME parameter, 111
- GEN parameter, 68, 111
- GTNAME parameter, 66, 70

## I

- IDTNAME parameter, 112
- IDTUGRP parameter, 112
- IMQNAME parameter, 69
- IMSGDEST parameter, 68
- installation

for z/VSE, 5

## L

length of user data passed to user exit 4, 112

LGBLSET macro

- ADL parameter, 109
- AVB parameter, 109
- CITSNM parameter, 110
- COR parameter, 110
- DBSVCTN parameter, 110
- DYNDBSVC parameter, 111
- ENTPT parameter, 111
- GBLNAME parameter, 111
- GEN parameter, 111
- IDTNAME parameter, 112
- IDTUGRP parameter, 112
- LOGID parameter, 112
- LUINFO parameter, 112
- LUSAVE parameter, 112
- LX1NAME parameter, 113
- LX2NAME parameter, 113
- modifying, 108
- MRO parameter, 113
- NETOPT parameter, 113
- NTGPID parameter, 114
- NUBS parameter, 114
- OPSYS parameter, 114
- PARMTYP parameter, 115
- PRE parameter, 115
- PURGE parameter, 115
- RENT parameter, 116
- RETRYX parameter, 116
- REVID parameter, 116
- REVIEW parameter, 116
- REVEL parameter, 117
- RMI parameter, 117
- RTXNAME parameter, 117
- RVCLNT parameter, 117
- SAF parameter, 118
- SAP parameter, 118
- SAPSTR parameter, 118
- SVCNO parameter, 118
- TPMON parameter, 119
- TRUENM parameter, 119
- UBPLOC parameter, 119
- UBSTIME parameter, 120
- UBTYPE parameter, 120
- UES parameter, 121
- USERX1 parameter, 121
- USERX2 parameter, 121
- XWAIT parameter, 122

link globals module name, 111

link globals table, 77

- LOGID parameter, 112
- LUINFO parameter, 112
- LUSAVE parameter, 112
- LX1NAME parameter, 113
- LX2NAME parameter, 113

## M

MACINS macro

- description, 66

- example, 66

- syntax, 66

MACIOPT macro

- ENTRY=FINAL statement, 70

- ENTRY=GLOBAL statement, 68

- ENTRY=GROUP statement, 69

- example, 70

- syntax, 67

macros

- MACINS, 66

MAX-AC-IO parameter, 150

MAXANZ parameter, 150

MDBSVC macro

- parameters, 105

- statement types, 104

- TYPE=FINAL statement syntax, 105

- TYPE=GEN statement syntax, 105

- TYPE=INIT statement syntax, 105

- using, 103

MNTRUE parameter, 69

MRO parameter, 113

multiple CICS TRUE support

- overview, 64

## N

Natural group ID, 114

NETOPT parameter, 113

NR-EXT parameter, 150

NR-PERCENT parameter, 151

NR1-N3 parameter, 150

NTGPID parameter, 114

NUBS parameter, 114

## O

operating system parameter, 114

OPSYS parameter, 106, 114

## P

parameter list area, 115

PARMTYP parameter, 115

PRE parameter, 115

PREFIX parameter, 106

PURGE parameter, 115

PURGE-UQE parameter, 151

## R

reentrant globals module flag, 116

RENT parameter, 116

retry command exit flag, 116

RETRYX parameter, 116

REVID parameter, 116

REVIEW parameter, 116

REVEL parameter, 117

RMI parameter, 117

routing Adabas calls, 99

RTXNAME parameter, 117

RVCLNT parameter, 117

## S

SAF parameter, 118  
 SAP ID string parameter, 118  
 SAP parameter, 118  
 SAP user ID generation support parameter, 118  
 SAPSTR parameter, 118  
 SAVEFDT parameter, 151  
 setting AOS defaults, 150  
 STATINTV parameter, 151  
 SVC parameter, 107  
 SVC routing  
   by database ID, 99  
 SVCNO parameter, 118

## T

TABNAME parameter, 107  
 target database ID  
   default, 112  
 task-related user exit (TRUE), 75  
 TCP/IP access (ADATCP)  
   enabling in z/VSE, 129  
 TID-DISPLAY parameter, 151  
 TIMELA parameter, 151  
 TIN-JOBN parameter, 151  
 TP monitors  
   CICS application stub, 65  
   CICS execution unit, 64  
   CICS installation options table, ACIOPT, 65  
   CICS names module, ACINAMES, 65  
   MACINS macro, 66  
 TP operating environment parameter, 119  
 TPMON parameter, 119  
 TRUENAME parameter, 66  
 TRUENM parameter, 75, 77, 119  
 TYPE=FINAL statement  
   MDBSVC macro, 104  
   syntax, 105  
 TYPE=GEN statement  
   MDBSVC macro, 104  
   syntax, 105  
 TYPE=INIT statement  
   MDBSVC macro, 104  
   syntax, 105

## U

UBPLOC parameter, 119  
 UBSTIME parameter, 120  
 UBTYP parameter, 120  
 UES parameter, 121  
 Universal Encoding Support (UES)  
   enabling in z/VSE, 123  
 universal encoding support parameter, 121  
 user block  
   pool allocation parameter, 119  
   scan time parameter, 120  
   type parameter, 120  
 user blocks created by CICS link routine, 114  
 user exit 1 flag, 121  
 user exit 1 module name, 113  
 user exit 2 flag, 121  
 user exit 2 module name, 113

user exit 4  
   length of user data passed, 112  
 user exits  
   AOSEX1, 150  
 user save area for LUEXIT1 and LUEXIT2, 112  
 USERX1 parameter, 121  
 USERX2 parameter, 121

## X

XWAIT parameter, 122

## Z

z/VSE  
   changing logical units, 142  
   enabling direct TCP/IP (ADATCP) access, 129  
   enabling UES support, 123  
 zaps  
   for changing z/VSE logical units, 142

