

# **webMethods EntireX**

## **Installation under z/OS**

Version 10.9

April 2023

This document applies to webMethods EntireX Version 10.9 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 © 1997-2023 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: EXX-INSTALL-109-20230403ZOS**

## Table of Contents

Installing EntireX under z/OS .....	v
1 About this Documentation .....	1
Document Conventions .....	2
Online Information and Support .....	2
Data Protection .....	3
2 z/OS Prerequisites .....	5
3 Introduction to Installing EntireX under z/OS .....	7
Installation Method .....	8
Licensing Considerations .....	8
Installable Components .....	9
Contents of Mainframe Installation Medium .....	9
Installation Jobs .....	11
Storage Requirements .....	11
Copying the Contents of the Installation Medium to Disk .....	11
4 Simplified z/OS Installation Method .....	15
Overview .....	16
Delivered Members .....	16
Installation Parameters .....	19
Installation Jobs .....	22
Installation Verification .....	24
Administration Tasks .....	24
5 Installing EntireX Broker under z/OS .....	27
Step 1: Set up the Attribute File .....	28
Step 2: Edit the Broker Startup Procedure .....	28
Step 3: Install the Broker Stubs and Bind the Broker Stub Executables .....	29
Step 4: Authorize and Assign the Broker STEPLIB Data Sets .....	30
Step 5: Configure Time Zone Settings .....	31
Step 6: (Optional) Define the Persistent Store .....	31
6 Installing the EntireX RPC Servers under z/OS .....	33
Installing the RPC Server for CICS .....	34
Installing the RPC Server for Batch .....	39
Installing the RPC Server for IMS .....	41
Installing RPC Examples for z/OS .....	44
7 EntireX CICS® Socket Listener .....	45
8 Installing EntireX Security under z/OS .....	47
Installing EntireX Security for Broker Kernel .....	48
Setting up EntireX Security for Broker Stubs .....	51
9 Installing EntireX Java Components under z/OS UNIX .....	55
Scope .....	56
Installation Steps .....	56
Configuring and Administering the Java Components .....	57
10 Installing the EntireX ICU Custom Converter Build Environment under z/OS UNIX .....	59

Installation Steps .....	60
11 Verifying the z/OS Installation .....	61
EntireX Broker .....	62
Sample Programs for Client (BCOC) and Server (BCOS) .....	62
RPC Server for CICS .....	63
RPC Server for Batch .....	63
RPC Server for IMS .....	63
12 Installing Adabas Components for EntireX under z/OS .....	65
Initializing the Adabas Communication Environment .....	66
SVC Integrity Validation .....	76
Requirements for Cross-Memory Services .....	77
Applying Zaps .....	78
13 Installing Adabas with TP Monitors .....	81
Preparing Adabas Link Routines for IBM Platforms .....	82
Installing Adabas with IMS TM under Adabas 8 .....	84
General Considerations for Installing Adabas with CICS .....	86
Installing Adabas with CICS under Adabas 8 .....	88
Installing Adabas with Com-plete under Adabas 8 .....	103
General Considerations for Installing Adabas with Batch/TSO .....	105
Installing Adabas with Batch/TSO under Adabas 8 .....	107
Modifying Source Member Defaults (LGBLSET Macro) in Version 8 .....	109

---

# Installing EntireX under z/OS

---

There are two methods for installing EntireX under z/OS:

- the simplified installation method; this is the method we recommend
- the classic installation method as used in previous versions of EntireX (prior to version 9.9)

The information you need depends on the installation method you choose. The following sections apply to both methods:

<a href="#">Prerequisites</a>	Prerequisites
<a href="#">Introduction</a>	Installation scope; contents of the mainframe installation medium; copying the contents to disk.

The following section applies to the simplified method only:

<a href="#">Simplified Installation</a>	New, simplified installation procedure. All JCL members are now in library EXX109.JOBS. This data set has been thoroughly restructured; it contains installation, installation verification and maintenance jobs of all EntireX subproducts. This is the recommended installation method.
---	---

The following sections apply to the classic method only:

<a href="#">Installing EntireX Broker</a>	Describes the steps for installing EntireX Broker under z/OS.
<a href="#">Installing EntireX RPC Servers</a>	Describes how to install the EntireX RPC servers under CICS, Batch and IMS.
<a href="#">EntireX CICS® Socket Listener</a>	Describes how to install the CICS Socket Listener.

The following sections apply to both methods:

<a href="#">Verifying the Installation</a>	Describes how to verify whether the installation of EntireX was successful.
<a href="#">Installing EntireX Security</a>	Provides links to the steps required for installing EntireX Security under z/OS.
<a href="#">Installing EntireX Java Components</a>	How to install EntireX Java components under z/OS UNIX.
<a href="#">ICU Custom Converter</a>	How to install the EntireX ICU Custom Converter Build Environment under z/OS UNIX.
<a href="#">Installing Adabas Components for EntireX</a>	Installing the Adabas components for EntireX if you do not have Adabas already installed at your site and you want to use SVC communication.

<i>Installing Adabas with TP Monitors for EntireX</i>	Installing Adabas components for EntireX in batch mode and with its teleprocessing (TP) monitors.
---	---

Installation prerequisites are listed for all platforms centrally; see *z/OS Prerequisites* under *Prerequisites* in the EntireX Release Notes. See also *General Installation Information* for topics that apply to multiple operating systems.



**Note:** If you want to use EntireX on z/OS together with the Eclipse-based Designer components, you need to install the respective EntireX components under Linux or Windows, using the Software AG Installer. See *Software AG Installer* under *Software AG Suite & Cross-Product Guides*.

# 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:  Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies:  Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the   symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

## Online Information and Support

---

### Product Documentation

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

In addition, you can also access the cloud product documentation via <https://www.software-ag.cloud>. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

### Product Training

You can find helpful product training material on our Learning Portal at <https://knowledge.softwareag.com>.



## Tech Community

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

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

## Product Support

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

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

## Data Protection

---

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



## 2 z/OS Prerequisites

---



**Note:** The supported versions of CICS and IMS are listed under *IBM Supported Platforms* on the Software AG Corporate Website.

Component	Prerequisites
EntireX Broker with Transport Method NET	■ Adabas SVC version of the highest Adabas/WAL version on this platform in your environment. If in doubt, see your Adabas documentation regarding SVC compatibility.
COBOL RPC Client and Server	■ To compile the sources generated by the Designer component COBOL Wrapper: IBM Enterprise COBOL for z/OS 6.2 or 6.3.
PL/I RPC Client and Server	■ To compile the sources generated by the Designer component PL/I Wrapper: Enterprise PL/I for z/OS.
RPC Server and Listener for IBM® MQ	■ IBM® MQ 9.0 or 9.1.
Software AG Licensing	■ MLC version 1.3.6 or above. See <i>Software AG Licensing Messages</i> .



# 3

## Introduction to Installing EntireX under z/OS

---

■ Installation Method .....	8
■ Licensing Considerations .....	8
■ Installable Components .....	9
■ Contents of Mainframe Installation Medium .....	9
■ Installation Jobs .....	11
■ Storage Requirements .....	11
■ Copying the Contents of the Installation Medium to Disk .....	11

## Installation Method

---

There are two methods for installing EntireX under z/OS:

- the simplified installation method
- the classic installation method as used in previous versions of EntireX

The information provided in this section applies to both methods. Then continue with either [\*Simplified z/OS Installation Method\*](#) or [\*Installing EntireX Broker under z/OS\*](#).

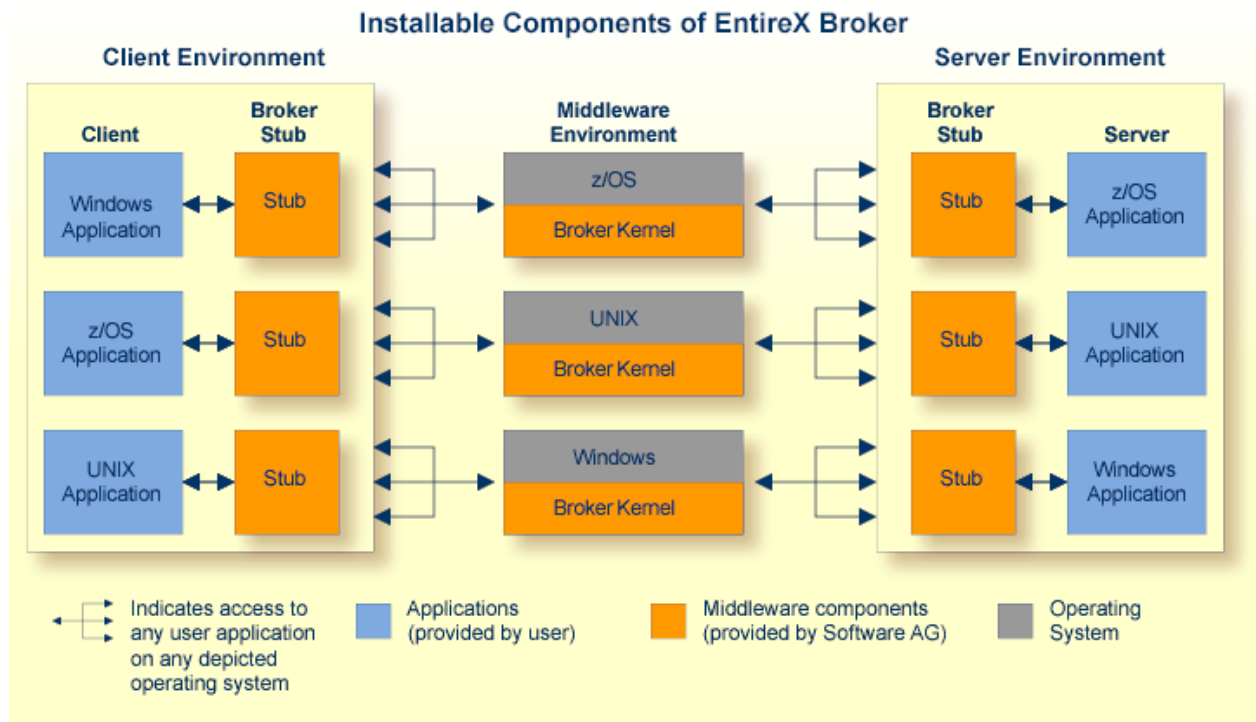
## Licensing Considerations

---

Software AG licensing requires that the modules `LICMAIN` and `LICUTIL` can be loaded when the EntireX Broker starts up. These modules are distributed in the `MLC136.LOAD` library.

See *z/OS Licensing Data Sets*.

## Installable Components



## Contents of Mainframe Installation Medium

The webMethods EntireX installation medium contains the data sets required to install all EntireX z/OS components. Data set names begin with a product code that identifies the module, as in the following tables.

With webMethods EntireX, the following additional products are included:

- Various Base Products for EBCDIC platforms
- Transport Services for EBCDIC platforms

Code	webMethods EntireX Component
EXP	EntireX RPC.
EXB	EntireX Broker.
EXX	Common modules used by all EntireX components.
MLC	Software AG's common mainframe license check software.
WAL	Adabas Limited Libraries.

The installation medium contains the data sets listed below:

Data Set Name	Description
EXB109.LOAD	Broker-specific load library (PDS/E).
EXB109.SRCE	Broker-specific source library
EXP109.EXPL	RPC compressed examples library (COBOL and PL/I).
EXP109.INCL	RPC include and copybook library.
EXP109.LB00	RPC batch load library.
EXP109.LD00	RPC CICS load library.
EXP109.MACS	RPC macros.
EXP109.SD00	RPC side deck library.
EXP109.SRCE	RPC source data set.
EXX109.CERT	Certificates to demonstrate SSL/TLS connections.
EXX109.DC00	Readme.
EXX109.LICS	License key.
EXX109.JOBS	This data set has been thoroughly restructured. It contains installation, installation verification and maintenance jobs of all EntireX subproducts.
EXX109.LOAD	Common load library (PDS/E), also includes broker stubs.
EXX109.SD00	Common side deck library.
EXX109.TAR	All Java components for z/OS. See <a href="#">Installing EntireX Java Components under z/OS UNIX</a> .
EXX109.SRCE	Common source library.
EXX109.ZAPS	Common zaps library.
MLC136.JOBS	Sample job library for Software AG's common mainframe license check software. <sup>(*)</sup>
MLC136.LOAD	Load library for Software AG's common mainframe license check software. <sup>(*)</sup>
WAL842.LOAD	Adabas limited load library. <sup>(*)</sup>
WAL842.SRCE	Adabas limited source library. <sup>(*)</sup>
WAL842.JOBS	Adabas limited jobs library. <sup>(*)</sup>

<sup>(\*)</sup> This was the latest version of the MLC and WAL components when this version of EntireX was originally released, and the version with which EntireX was tested. If a higher version of MLC or WAL becomes available at a later time, we recommend you use this.



## Installation Jobs

---

The installation of Software AG products on z/OS is performed by installation jobs. There are three possible scenarios:

- The jobs are the manually adapted sample jobs in the delivered JOBS data set.
- The jobs are generated by System Maintenance Aid (SMA). For each step of the installation procedure, an installation job is generated by SMA according to your specifications in SMA. If you are not using SMA, follow the instructions using your own jobs.
- You are using the simplified installation method. See [Simplified z/OS Installation Method](#).

Information on using SMA for the installation process is provided in the *System Maintenance Aid Manual*.

## Storage Requirements

---

For specific storage requirements, see the Software AG Product Delivery Report.

## Copying the Contents of the Installation Medium to Disk

---

### Installation Steps

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 all data sets from tape to disk.

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

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

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

- Modify the following sample job according to your requirements:

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

where:

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

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

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

- Modify *hilev*.COPY.JOB according to your requirements:

Set EXPDT to a valid expiration date, for example, 99365.

Set HILEV to a valid high-level qualifier, for example, USERLIB.

Set LOCATION to a storage location, for example, STORCLAS=ABC or UNIT=3390,VOL=SER=USR123.

### Step 3: Submit COPY.JOB

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

### Next Steps

Continue the installation with either [Simplified z/OS Installation Method](#) or [Installing EntireX Broker under z/OS](#).



# 4

## Simplified z/OS Installation Method

---

■ Overview .....	16
■ Delivered Members .....	16
■ Installation Parameters .....	19
■ Installation Jobs .....	22
■ Installation Verification .....	24
■ Administration Tasks .....	24

## Overview

---

The simplified installation requires the following steps:

1. Set the installation keyword parameters in a parameter member.
2. Execute a REXX script that updates the delivered installation jobs with the values of that parameter member.
3. Auto-submit the updated jobs if requested in the parameter file, or make them ready for a manual submission.

## Delivered Members

---

The delivered members in the original `EXX109.JOBS` data set are:

`#IPARMS`

The installation parameters.

`#RX#CMD`

The REXX update script.

`#RX#JOB`

A job to execute `#RX#CMD` in batch.

`#INSTALL`

Generated on first submission: Documents the installation jobs required as a result of the components selected.



**Note:** Only the `#INSTALL` job will be copied to the target `.JOBS` data set. The first three members remain in the original `EXX109.JOBS` data set.

### #IPARMS - The Parameters

This member keeps all parameters relevant for installation and is located and maintained in the delivered data set `EXX109.JOBS` only, which was copied directly from the installation medium. To avoid a complete setup, you can select an already maintained `#IPARMS`. See [#RX#JOB- A REXX Batch Job](#).

## Member Layout

The parameter member contains the following sections:

1. global installation parameters
2. job card related parameters
3. installable unit subproduct selection parameters
4. installable unit installation parameters
5. data set names / high-level qualifiers
6. VSAM file characteristics

## Parameter Line Layout

A parameter member line has the following layout:

```
* ----- *
```

```
* JOBCARD
```

```
* ----- *
```

```
keyword    = value          * comment
```

where *keyword* is a placeholder for a keyword

= is the value separator

*value* is the keyword value

\* introduces a comment. A line that begins with an asterisk in column 0 is treated as a comment line.

## #RX#CMD- The REXX Script

This script copies all the necessary jobs from the original *EXX109.JOBS* data set of the delivered installation medium to a selected installation *target.JOBS* data set. All placeholders in the original jobs will be replaced with the values set in the parameter member. An existing *target.JOBS* data set of a previous generation will be saved in a GDG data set first.

A *#INSTALL* member will be created for documentation and to distinguish between the jobs required as a result of the selected components and the optional installation jobs. All required installation jobs can be submitted directly with this member.

## #RX#JOB- A REXX Batch Job

Calls the #RX#CMD REXX script from batch.

The #RX#CMD script expects six KEYWORD parameters that need to be maintained and set to suitable values in the #RX#JOB first. There are two groups of keyword parameters: mandatory and optional.

### ■ Mandatory Keyword Parameters

- ORIG - The name of the EXX109.JOBS data set copied from the installation medium
- INST - The name of the target.JOBS data set that will contain the generated installation jobs

### ■ Optional Keyword Parameters

- PARM - The name of a data set containing an already maintained #IPARMS member. If a #IPARMS member is found here it will be selected for the installation jobs generation process.
- MSGS - Switch IEBCOPY sysout messages on or off. Possible values are:
  - Y - Show IEBCOPY sysout (default)
  - N - Hide IEBCOPY sysout
- INST\_SMS - Switch SMS control on or off for the target.JOBS data set named with the mandatory INST keyword parameter. Possible values:
  - Y - Switch ON the SMS controlled allocation of the installation jobs data set
  - N - Switch OFF the SMS controlled allocation of the installation jobs data set
- INST\_VOL - Contains the SMS class name or the VOLSER name depending on the value of keyword parameter INST\_SMS
  - INST\_SMS=Y - The SMS class name
  - INST\_SMS=N - The DASD VOLSER name



**Important:** It is mandatory to edit this job *manually* before submission to define at least the mandatory keywords naming the input (*EXX109.JOBS*) and output (*target.JOBS*) data sets.

## #INSTALL - Document the Required Installation Jobs (Depending on Selected Components)

This member will be generated in both, the *EXX109.JOBS* and the *target.JOBS* data sets. It has two functions:

1. **Documentation of the Installation Jobs Required as a Result of the Selected Installable Units**  
Each required installation job is represented here with a separate line headed by its member name in the *EXX109.JOBS*.
2. **Submission of the Required Installation Jobs**  
As this member is a job itself, it can be submitted to submit all the required installation jobs. You can also select a subset of jobs for submission by (un)commenting their execution lines.



## Installation Parameters

This section lists the parameters available with the simplified installation procedure:

- [Installable Unit Selection Parameters](#)
- [Job Replacement Parameters](#)

### Installable Unit Selection Parameters

Use these parameters to determine the scope of the installation.

Installable Unit	Description
InstBroker	Subproduct installation: Select the Broker (Y/N).
InstRpcBatch	Subproduct installation: Select the RPC Server for Batch (Y/N).
InstRpcCics	Subproduct installation: Select the RPC Server for CICS (Y/N).
InstRpcIms	Subproduct installation: Select the RPC Server for IMS (Y/N).
InstSSL	Subproduct installation: Select the SSL libraries (Y/N).

### Job Replacement Parameters

Parameter Type	Parameter Name	Job Replacement Tag	Description
Global Parameters	WorkingStorUnit	<unit>	Temporary storage unit.
	MaxBackups	<MaxBackups>	Upper limit of GDG data set.
	HLQ-TapeKit	<Tape>	High-level qualifier of the JOBS data sets copied to DASD from the delivered installation medium.
	HLQ-InstallKit	<Inst>	High-level qualifier of the data set containing the generated installation jobs.
	Submit	<,>	Autosubmit (Y/N) the mandatory installation jobs for the selected units to the HOLD queue.
Job Card Related	JobName	<jobname>	Job name.
	Accounting	<account>	Accounting information.
	Programmer	<name>	Programmer information.
	Notify	<notify>	Notification user ID.
	Region	<region>	Storage amount.
	Priority	<prio>	Selection priority.
	Time	<time>	Max. CPU time.
	MsgLevel	<msglvl>	Job output level.

Parameter Type	Parameter Name	Job Replacement Tag	Description
	JobClass	<class>	Run class.
	MsgClass	<msgclass>	Message class.
	Special	<special>	Special job info (e.g. for JES exits).
High-level Qualifiers	HLQ-Adabas	<ADAvrs>	High-level qualifier of the installed Adabas.
	HLQ-AdabasIndi	<WALvrs>	High-level qualifier of the installed Independent Adabas.
	HLQ-AdabasDB	<ADAdb>	High-level qualifier of the target Adabas database for the ADA/NAT update jobs.
	HLQ-Natural	<NATvrs>	High-level qualifier of the installed Natural.
	HLQ-EntireX	<EXXvrs>	High-level qualifier of the installed EntireX Global.
	HLQ-Broker	<EXBvrs>	High-level qualifier of the installed EntireX Broker.
	HLQ-RpcServer	<EXPvrs>	High-level qualifier of the installed EntireX RPC Servers.
	HLQ-Cics	<CICS>	High-level qualifier of the CICS to be used for installation.
	HLQ-CicsCSD	<CICSCSD>	High-level qualifier of the CICS CSD to be used for installation.
	HLQ-IMS	<IMS>	High-level qualifier of the IMS to be used for installation.
	HLQ-MQM	<MQM>	High-level qualifier of the MQSeries data sets.
	HLQ-COB	<COB>	High-level qualifier of the COBOL compiler data sets.
	HLQ-PLI	<PLI>	High-level qualifier of the PL/I compiler data sets.
	HLQ-CEE	<CEE>	High-level qualifier of the CEE language environment data sets.
Broker-related	BrokerID	<BrokerID>	EntireX Broker name for TCP/IP or Entire Net-Work access.
	BrokerNode	<BrokerNode>	EntireX Broker node number for Entire Net-Work access.
	MigDsnBrokerV72	<EXBmig>	Broker source data set for automated parameter migration.
	MigDsnEntireXV72	<EXXmig>	EntireX source data set for automated parameter migration.
	MigOldAttrMember	<OldAttrMember>	Migration: Member name of old attributes (EXBATTR).

Parameter Type	Parameter Name	Job Replacement Tag	Description
	MigOldCommMember	<OldCommMember>	Migration: Member name of old TCP/IP parameters (EXBCOMM).
	MigOldStorMember	<OldStorMember>	Migration: Member name of old persistent storage parameters (PSFFP).
	MigOldParmMember	<OldParmMember>	Migration: Member name of old parameters (EXBPARM).
	MigNewAttrMember	<NewAttrMember>	Migration: Member name of new consolidated attributes.
RPC Server-related	IMS-PsbName	<ImsPsbName>	IMS RPC: PSB name.
	DSN-ImsLoad3GL	<DSNImsLoad3GL>	IMS RPC: COBOL and PL/I server load modules data set.
	DSN-BatLoad3GL	<DSNBatLoad3GL>	Batch RPC: COBOL and PL/I server load modules data set.
	DSN-DfhLoad3GL	<DSNDfhLoad3GL>	CICS RPC: COBOL and PLI/I server load modules data set.
	DSN-ImsStub3GL	<DSNImsStub3GL>	IMS RPC: COBOL and PL/I stub load modules data set.
	DSN-BatStub3GL	<DSNBatStub3GL>	Batch RPC: COBOL and PL/I stub load modules data set.
	DSN-DfhStub3GL	<DSNDfhStub3GL>	CICS RPC: COBOL and PL/I stub load modules data set.
	HLQ-RpcCicsSVM	<HlqDfhSvmFile>	High-level qualifier of the RPC Server for CICS server-side mapping container (VSAM file). See Note 1,2,3.
	HLQ-RpcSVM	<HlqSvmFile>	High-level qualifier of the RPC Server for Batch or IMS server-side mapping container (VSAM file). See Note 1,2,3.
	RPC-ImsClass	<ImsClass>	IMS RPC: Class name of service triplet.
	RPC-ImsServer	<ImsServer>	IMS RPC: Server name of service triplet.
	RPC-ImsService	<ImsService>	IMS RPC: Service name of service triplet.
Adabas / Natural-related Parameters	AdaSvcNo	<AdaSvcNo>	Adabas router SVC number.
	AdaDeviceType	<AdaDeviceType>	Adabas database device type.
	AdaDBID	<AdaDBID>	Adabas database ID in the router SVC.
	NatBatch	<NatBatch>	Name of the Natural batch nucleus.
	Fnat	<Fnat>	Adabas file number of the Natural FNAT.
	Fuser	<Fuser>	Adabas file number of the Natural FUSER.
	Fdic	<Fdic>	Adabas file number of the Natural FDIC.
	AdaFileNoExbDiv	<DivFileNo>	Adabas file number of the Broker persistent store.

Parameter Type	Parameter Name	Job Replacement Tag	Description
VSAM file allocation characteristics for server-side mapping container (see Note 1,2,3)	RLS-SVM	<rls>	Record Level Sharing (Yes or No).
	SMS-SVM		SMS managed (Yes or No).
	VolSer-SVM	<sms>	SMS storage class name or VOLSER name (depending on SMS-SVM).
Broker VSAM file allocation characteristics for persistent store linear cluster	LLQ-EXB	<llq-exb>	Last-level qualifier of the Broker VSAM cluster.
	SMS-EXB		SMS managed (Yes or No).
	VolSer-EXB	<sms-exb>	SMS storage class name or VOLSER name (depending on SMS-EXB).

**Notes:**

1. If you have never used server-side mapping files (Designer files with extension .svm) from previous EntireX versions or you are a new customer, do not set up a server-side mapping container.
2. If you are using server-side mapping files (Designer files with extension .svm) at runtime, you need to set up a server-side mapping container.
3. Server mapping files with extension .svm are no longer supported at design time by the Designer. You can still use them at runtime in a server-side mapping container. All special COBOL syntax and features supported by server mapping files with extension .svm are also covered by server mapping files with extension .cvm. See *When is a Server Mapping File Required?*  
We recommend migrating .svm files to .cvm files. See *Migrating Server Mapping Files* under *Server Mapping Files for COBOL* in the Designer documentation.

## Installation Jobs

All existing jobs remain unchanged in the original *EXX109.JOBS* data set. Only the jobs necessary for the installation of the selected installable units will be copied to and updated within the designated *target.JOBS* data set by the *#RX#CMD* REXX script.



**Note:** This update process is not mandatory. All jobs can still be manually adapted as before. But to be able to successfully execute the installation script afterwards, the original *EXX109.JOBS* data set should be kept unchanged.

This section covers the following topics:

- [Generation Data Sets](#)
- [Preparation and Execution](#)

- Generation Process
- Submission

## Generation Data Sets

After a successful generation, the following data sets are in the system:

- **EXX109.JOBS**  
The delivered product data set. The members of this data set will remain unchanged with the exception of the #IPARMS parameters and possibly the #RX#JOB.
- **target.JOBS**  
The target data set which will contain the selected and updated jobs from the *EXX109.JOBS* data set.
- **target.JOBS.BAK**  
A VSAM GDG base catalog entry for the GDG data sets.
- **target.JOBS.GnnnnVnn**  
With any subsequent generation, all jobs of the previous generation in the *target.JOBS* data set are kept here until the maximum value set in the MAXBACKUPS parameter (to be found in #IPARMS) is reached.

## Preparation and Execution

### ➤ To prepare and install the installation jobs



**Note:** It is mandatory to edit the job *manually* first to set the input and output JOBS data set names.

- 1 Update the *EXX109.JOBS* (#IPARMS) parameter member.
- 2 Submit the job *EXX109.JOBS*(#RX#JOB) to execute the REXX script #RX#CMD.

## Generation Process

Every keyword parameter has a well defined default value. This default will be replaced only if requested by the parameter member, that is, if a KEYWORD=VALUE entry is successfully identified. When a keyword is deleted from the parameter member (or commented out), this default will always be in place.

## Submission

### ➤ To submit the installation jobs

- In the #IPARMS member select the switch SUBMIT.

- If set to "Y":

Any job is submitted with a TYPRUN=HOLD job card parameter.



**Note:** This allows a final check to ensure a correct generation. Any job can then be released manually one after the other in the numbered order or can be cancelled if an error was encountered.

If you cannot issue the JES release command for security reasons, you can submit using the generated #INSTALL job in the target.JOBS instead.

- If set to "N":

The TYPRUN=HOLD job card parameter is set to comment and no job is submitted at all.



**Note:** The #INSTALL member will be generated regardless of the SUBMIT parameter value.

## Installation Verification

---

Installation verification is the same for both installation methods. See [Verifying the z/OS Installation](#).

## Administration Tasks

---

After installation has been completed, various administration tasks may be necessary.

- [Modify Broker Attribute File](#)
- [Set up Broker Stubs](#)
- [Define the Persistent Store](#)

- [Setting up the EntireX RPC Servers](#)

## **Modify Broker Attribute File**

Customize the attribute settings to suit your needs. See *Broker Attributes*.

## **Set up Broker Stubs**

See *Administering Broker Stubs*.

## **Define the Persistent Store**

A persistent store can be optionally used for storing unit of work messages and message status information to disk. For z/OS, you can use an Adabas persistent store (recommended) or a DIV persistent store that uses a VSAM linear data set.

### **Adabas Persistent Store**

See *Implementing an Adabas Database as Persistent Store* and *Adabas-specific Attributes* in the Broker attribute file documentation for more information.

### **DIV Persistent Store**

See *Implementing a DIV Persistent Store* and *DIV-specific Attributes* for more information.

## **Setting up the EntireX RPC Servers**

See *RPC Server for CICS*, *RPC Server for Batch* or *RPC Server for IMS* for more information.





# 5

## Installing EntireX Broker under z/OS

---

■ Step 1: Set up the Attribute File .....	28
■ Step 2: Edit the Broker Startup Procedure .....	28
■ Step 3: Install the Broker Stubs and Bind the Broker Stub Executables .....	29
■ Step 4: Authorize and Assign the Broker STEPLIB Data Sets .....	30
■ Step 5: Configure Time Zone Settings .....	31
■ Step 6: (Optional) Define the Persistent Store .....	31

This chapter describes the steps for installing EntireX Broker under z/OS.

## Step 1: Set up the Attribute File

---

(SMA Job I070 / Step 7604)

The EXB109.SRCE data set contains a sample Broker attribute file (member EXBATTR). Copy this member to a member of your choice and customize the attribute settings to suit your needs. The file must then be allocated to DD name ETBFILE in the Broker Startup Procedure (see step [Step 2: Edit the Broker Startup Procedure](#)). For detailed information about the attributes, see *Broker Attributes*.

## Step 2: Edit the Broker Startup Procedure

---

(SMA Job I070 / Step 7606)

Copy the example member EXBSTART (the EntireX Broker Startup Procedure) from the EXX109.JOBS data set to your proclib and edit it to suit your naming conventions.

### Note on Region Size

We recommend setting the region size for the Broker to "0M". This means the operating system allocates resources as required. If you wish to influence the resources allocated, you can also set the size depending on the amount of fixed virtual memory that is allocated during Broker initialization. This value is displayed in diagnostic message ETBD0284 of the Broker trace output. Add to this value an additional 64M to account for other storage possibly needed during the Broker execution.

## Overview of DD Names

The following table describes the DD names used in this file:

DD Name	Description	See also
CEEDUMP	Exception handling will write any dump data to this data set.	
DDCLOGR1	First dual command log data set.	<i>Command Logging in EntireX.</i>
DDCLOGR2	Second dual command log data set.	<i>Command Logging in EntireX.</i>
ETBCREP	Configuration report provides the contents of the variable definition file (optional) and the contents of the attribute file (required).	
ETBFILE	Broker attribute file.	<i>Step 1: Set up the Attribute File.</i>
ETBLIB	All load libraries required for execution of Broker.	
ETBLREP	License report provides the contents of the license file and some machine data.	
ETBMREP	Module report provides a list of all members in the STEPLIB data set concatenation.	
ETBPREP	Persistent store report provides a list of all records added to or deleted from the persistent store.	
ETBVAR	EXBVAR file.	<i>Step 1: Set up the Attribute File.</i>
LICENSE	EntireX License Certificate File.	
STORE01	Persistent store data sets file. Defines the DIV persistent store data set. This DD name is derived from the value specified in the persistent store format parameter DDNAME.	<i>Step 6: (Optional) Define the Persistent Store.</i>
ABNLIGNR	Needed to suppress ABEND-AID™ dumps.	

## Step 3: Install the Broker Stubs and Bind the Broker Stub Executables

The broker stub executables are distributed prelinked with unresolved references to z/OS components. To resolve these external references, relink them in your environment. Execute job EXBINST located in the EXX109.JOBS data set.

If you will be using the Broker stubs on z/OS, see *Administering Broker Stubs* in the z/OS Administration documentation.

## Step 4: Authorize and Assign the Broker STEPLIB Data Sets

---

Perform the following steps as required:

- [Step 4a: Authorize the EXB Data Set](#)
- [Step 4b: Assign all STEPLIB Data Sets](#)

### Step 4a: Authorize the EXB Data Set

The EXB data set in the STEPLIB DD needs to be APF-authorized in the following situations:

- Broker running with SECURITY=YES
- Broker using the DIV persistent store facility
- NET transport method

If SECURITY=YES is defined and the EXB data set is not APF-authorized, the broker will not initialize and error ETBE0090 is issued.



**Note:** Since EntireX 10.9 this is the only data set that needs to be APF-authorized.

### Step 4b: Assign all STEPLIB Data Sets

Assign *all* load libraries via DD name ETBLIB:

- EXB109.LOAD
- EXX109.LOAD
- WAL842.LOAD
- MLC136.LOAD

EntireX Broker runs in 64-bit addressing mode (AMODE). The Language Environment Runtime Libraries must be defined either in the LNKLIST of the system or as STEPLIB data sets:

- hlq.SCEERUN2
- hlq.SCEERUN

## Step 5: Configure Time Zone Settings

Perform the following to configure time zone settings:

- [Step 5a: Give Broker Access to the /etc/profileFile](#)
- [Step 5b: Verify Correct Time Zone Setting for Broker](#)

### Step 5a: Give Broker Access to the /etc/profileFile

Ensure that your security product (e.g., RACF, ACF/2) permits the Broker kernel to have read access to the */etc/profile* file.

The Broker kernel obtains the time zone setting from the TZ environment variable in the */etc/profile* file, if possible.



**Note:** If the Broker kernel is not allowed read access, daylight savings time (if applicable) will not be automatically recognized in the Broker kernel log messages. A message from the security product will indicate the lack of permission. Example:

```
ICH408I USER(QEUSALT1) GROUP(QADEPT  ) NAME(QEUSALT1 STC      )
      /etc/profile CL(FSOBJ  ) FID(01E2E8E2D9C4F500171E000001170000)
      INSUFFICIENT AUTHORITY TO OPEN
      ACCESS INTENT(R--) ACCESS ALLOWED(OTHER      ---)
      EFFECTIVE UID(0000999999) EFFECTIVE GID(00000000006)
```

### Step 5b: Verify Correct Time Zone Setting for Broker

Verify that the time zone setting in the TZ environment variable of the */etc/profile* file is correct for your site, since the Broker obtains the time zone setting from this file, if possible.

## Step 6: (Optional) Define the Persistent Store

The persistent store can be optionally used for storing unit of work messages and message status information to disk.



**Important:** For z/OS, the available persistent store types are Adabas file (perform step “a” below) and DIV file implemented with a VSAM linear data set (perform step “b” below).

See *Managing the Broker Persistent Store* to understand how the persistent store is implemented.

## Step 6a: Define a New Adabas Persistent Store

### Define the Adabas Persistent Store

(SMA Job I050 / Steps 7600 / 7610)

Use this step if you have not already defined an Adabas persistent store.

The Adabas persistent store driver is contained within the regular Broker load module. It is activated by specifying attribute `PSTORE-TYPE=ADABAS`. Use the supplied job EXBJ015 from data set EXX109.JOBS to define and install the persistent store file in your Adabas database. This job creates and loads the Adabas file into the database.

See *Implementing an Adabas Database as Persistent Store* and *Adabas-specific Attributes* for more information.

## Step 6b: Allocate a DIV Persistent Store

To allocate the persistent store, you will need to define a VSAM linear data set (LDS) using the IBM utility IDCAMS. See *Implementing a DIV Persistent Store* for details on creating the persistent store data sets.

The file containing the persistent store data set must then be allocated to DD name STORE01 in the Broker Startup Procedure (see Step [Step 2: Edit the Broker Startup Procedure](#)).

Use the supplied job EXBIDCAMS from data set EXX109.JOBS to define a VSAM linear data set for the persistent store.



**Note:** You must allocate a unique persistent store (if used) per Broker.

See *Implementing a DIV Persistent Store* for more information.

See also *DIV-specific Attributes*.

# 6

## Installing the EntireX RPC Servers under z/OS

---

■ Installing the RPC Server for CICS .....	34
■ Installing the RPC Server for Batch .....	39
■ Installing the RPC Server for IMS .....	41
■ Installing RPC Examples for z/OS .....	44

For Natural RPC servers, see *Setting Up a Natural RPC Environment* in your Natural documentation.

## Installing the RPC Server for CICS

---

The EntireX RPC Server for z/OS CICS® allows standard RPC clients to communicate with RPC servers on the operating system z/OS under CICS. It supports the programming languages COBOL and PL/I.

- [Updating the CICS Tables](#)
- [Modifying the CICS Startup JCL](#)
- [Binding the RPC Executables](#)
- [Build the ERXMAIN Control Block](#)
- [TCP/IP-enabling the CICS Region \(Optional\)](#)
- [Installing the Server-side Mapping Container for an RPC Server for CICS \(Optional\)](#)
- [Starting the EntireX RPC Server Automatically on CICS Startup \(Optional\)](#)
- [Stopping the EntireX RPC Server Automatically on CICS Shutdown \(Optional\)](#)
- [Installing Multiple EntireX RPC Servers in the same CICS \(Optional\)](#)
- [Using SSL/TLS Connections with the RPC Server for CICS \(Optional\)](#)

Prerequisites for all EntireX components are described centrally. See *z/OS Prerequisites*.



**Note:** Installing the RPC Server for CICS also installs the CICS Socket Listener used by the EntireX Adapter or RPC Server for CICS Socket Listener. For more information and configuration details see *Preparing for CICS Socket Listener* (EntireX Adapter | RPC Server for CICS Socket Listener).

### Updating the CICS Tables

EntireX RPC requires a number of enhancements to the CICS tables. These changes are usually performed by a CICS system programmer using standard system jobs.

- Execute step "Install CICS CSD defs for EntireX RPC Server" of job EXPINSTA located in the EXX109.JOBS data set. This will install the RPC Server for CICS with all relevant CSD entries.
- Tracing: TD queue entries for RPC server trace output may optionally be updated by uncommenting the DFHTRACE exec line. See also *ERXMAIN Macro* parameter TRC1.



## Transaction Definitions

```
DEFINE TRANSACTION(ERXM) GROUP(ERX) PROGRAM(ERXMAINT) TWASIZE(128)
DEFINE TRANSACTION(ESRV) GROUP(ERX) PROGRAM(RPCSRVC) TWASIZE(128)
DEFINE TRANSACTION(EC01) GROUP(ERX) PROGRAM(SQRECLT) TWASIZE(28)
```

Transaction ERXM is used to run the *RPC Online Maintenance Facility*.

Transaction ESRV is used to run the RPC Server for CICS; see *Customizing the RPC Server*.



### Notes:

1. If you have not extended your DCT (that is, you are using the default values), you must specify the option DCT= in the CICS SYSIN file.
2. Check also if you need to alter *IBM LE Runtime Options* - for example AMODE24, how to trap ABENDS etc.
3. If required, adapt CICS settings, for example TWASIZE. See *CICS Settings*.

## Modifying the CICS Startup JCL

The startup JCL for CICS must be modified to include the EXP109.LD00 and EXX109.LOAD data sets in the DFHRPL data set concatenation. This is to enable CICS to find the various programs that have been defined in the PPT.

You may also include a DD statement for ERXOUT extra partition data set. Once the startup JCL has been modified, restart your CICS system.

## Binding the RPC Executables

The RPC executables are distributed prelinked with unresolved external references to z/OS components. To resolve these external references, you need to relink them in your environment. Execute step "Link the RPC Server for CICS modules" of job EXPINSTA located in the EXX109.JOBS data set.

## Build the ERXMAIN Control Block

The ERXMAIN control block holds the RPC Server for CICS parameter settings. This configuration can be manually maintained by using the *RPC Online Maintenance Facility*.

A fully linked control block named ERXMAIN is delivered in the load library EXP109.LOAD. To alter this configuration, perform the following steps:

1. Adapt the RPC server configuration in the Assembler program EMAINGEN located in the EXP109.SRCE data set to the customer environment, see *Customizing the RPC Server*.
2. Execute step "Build the ERXMAIN Control Block" of job EXPINSTA located in the EXX109.JOBS data set.

The name of the ERXMAIN control block may be altered. This allows you to run multiple RPC Server for CICS instances. See [Installing Multiple EntireX RPC Servers in the same CICS \(Optional\)](#).

### TCP/IP-enabling the CICS Region (Optional)

If you are using transport method TCP/IP, your CICS region must be enabled for TCP/IP. Refer to your CICS documentation for details.

### Installing the Server-side Mapping Container for an RPC Server for CICS (Optional)

You can skip this step if you are a new customer or have never used server-side mapping files (Designer files with extension .svm) from previous EntireX versions.

If you are using server-side mapping files (Designer files with extension .svm) at runtime, you need to set up a server-side mapping container.

Execute step "Allocate and (CSD) define SVM file" of the EXPINSTA job located in the EXX109.JOBS data set. After updating the SVMFILE variable to a suitable name, the following steps are performed:

- allocation of the required VSAM cluster
- initialization of the cluster with the first server mapping (VSAM record) that matches the CICS advanced channel container example DFHCON of the EXP109.DVCO data set. See *Client and Server Examples for z/OS CICS*.
- CICS CSD definition with the given name in the SVMFILE variable

Server mapping files with extension .svm are no longer supported at design time by the Designer. You can still use them at runtime in a server-side mapping container. All special COBOL syntax and features supported by server mapping files with extension .svm are also covered by server mapping files with extension .cvm. See *When is a Server Mapping File Required?*  
We recommend migrating .svm files to .cvm files. See *Migrating Server Mapping Files* under *Server Mapping Files for COBOL* in the Designer documentation.

See also *Server Mapping Files for COBOL* | *Server-side Mapping Files* | [Job Replacement Parameters](#).

### Starting the EntireX RPC Server Automatically on CICS Startup (Optional)

#### ➤ To start the RPC server automatically on CICS startup

- 1 Insert a new PLT entry DFHPLT TYPE=ENTRY, PROGRAM=ERXSTART.
- 2 Rebuild the PLT for CICS startup.



**Note:** The EXP109.LD00 CICS load library contains a precompiled ERXSTART module with the default settings of the COBOL source member ERXSTART in EXP109.SRCE

## Stopping the EntireX RPC Server Automatically on CICS Shutdown (Optional)

### ➤ To stop the RPC server automatically on CICS shutdown

- 1 Insert a new PLT entry `DFHPLT TYPE=ENTRY, PROGRAM=ERXSTOP` into your CICS shutdown table.
- 2 Rebuild the PLT for CICS shutdown.



**Note:** The EXP109.LD00 CICS load library contains a precompiled `ERXSTOP` module with the default settings of the COBOL source member `ERXSTOP` in `EXP109.SRCE`

## Installing Multiple EntireX RPC Servers in the same CICS (Optional)

### ➤ To install a second RPC server in the same CICS

- 1 Copy the default RPC Server for CICS transaction definition `ESRV` and give it a unique name, e.g. `ESR2`.

```
CEDA COPY TRANSACTION(ESRV) GROUP(ERX) TO(ERX2) AS(ESR2)
```

- 2 Copy the default RPC Server for CICS *ERXMAIN Control Block* and give it a unique name, e.g. `ERXMAIN2`.

```
CEDA COPY PROGRAM(ERXMAIN) GROUP(ERX) TO(ERX2) AS(ERXMAIN2)
```

- 3 Add the new group `ERX2` to the CICS autoinstall list.

```
CEDA ADD GROUP(ERX2) LIST(listname) AFTER(groupname)
```

- 4 Build a new *ERXMAIN Control Block* and give it the name created above, e.g. `ERXMAIN2`.

As a minimum, set the *ERXMAIN Macro* parameter `REPL` in the *ERXMAIN Control Block* to the new RPC server transaction ID created above, e.g. `REPL=ESR2`.

The second RPC Server for CICS can now be started manually (see *Starting the RPC Server* under *RPC Online Maintenance Facility*), or automatically on CICS startup.

### ➤ To start a second RPC server automatically on CICS startup

- 1 Copy the default RPC Server for CICS autostart definition `ERXSTART` and give it a unique name, e.g. `ERXSTR2`.

```
CEDA COPY PROGRAM(ERXSTART) GROUP(ERX) TO(ERX2) AS(ERXSTRT2)
```

- 2 "CEDA Install" the new autostart definition.
- 3 Modify the RPC Server for CICS PLT startup routine ERXSTART from data set EXP109.SRCE.
  - Update RPC-TRANSID with ESR2.
  - Update RPC-INPUT with MEM=ERXMAIN2.
- 4 Compile and link the modified source and give it the name defined above, e.g. ERXSTRT2.
- 5 Insert a new PLT entry DFHPLT TYPE=ENTRY, PROGRAM=ERXSTRT2.
- 6 Rebuild the PLT for CICS startup.

➤ **To stop a second RPC server automatically on CICS shutdown**

- 1 Copy the default RPC Server for CICS autostop definition ERXSTOP and give it a unique name, e.g. ERXSTOP2.

```
CEDA COPY PROGRAM(ERXSTOP) GROUP(ERX) TO(ERX2) AS(ERXSTOP2)
```

- 2 "CEDA Install" the new autostop definition.
- 3 Modify the RPC Server for CICS PLT routine ERXSTOP from data set EXP109.SRCE.
  - Update RPC-TRANSID with ESR2.
  - Update RPC-INPUT with MEM=ERXMAIN2.
- 4 Compile and link the modified source and give it the name defined above, e.g. ERXSTOP2.
- 5 Insert a new PLT entry DFHPLT TYPE=ENTRY, PROGRAM=ERXSTOP2 into your CICS shutdown table.
- 6 Rebuild the PLT for CICS shutdown.

**Using SSL/TLS Connections with the RPC Server for CICS (Optional)**

See *Using SSL/TLS with the RPC Server*.

## Installing the RPC Server for Batch

The EntireX RPC Server for z/OS Batch allows standard RPC clients to communicate with RPC servers on the operating system z/OS running in batch mode. It supports the programming languages COBOL, PL/I and C and works together with the *COBOL Wrapper* and *IDL Extractor for COBOL*.

- [Prepare Your Startup JCL](#)
- [Customize Your Server Configuration](#)
- [Binding the RPC Executables](#)
- [Using z/OS Privileged Services](#)
- [Installing the Server-side Mapping Container for an RPC Server for Batch \(Optional\)](#)
- [Using SSL/TLS Connections with the RPC Server for Batch \(Optional\)](#)

Prerequisites for all EntireX components are described centrally. See *z/OS Prerequisites*.

### Prepare Your Startup JCL

The RPC Server for Batch can run as a started task. The installation medium contains the following sample JCL:

■ EXPSRVB

#### > To prepare your startup JCL

- 1 Modify the example started task EXPSRVB of the EXX109.JOBS data set to suit your installation.
- 2 Modify the CONFIG DD statement to point to your server configuration file.
- 3 Concatenate your server application data set to the RPC server STEPLIB.
- 4 Add the EntireX RPC server JCL to your TASKLIB data set.

### Customize Your Server Configuration

Modify the sample parameter member CONFIG from EXP109.SRCE. The RPC Server for Batch is optimized for use in COBOL environments. Nevertheless, as a minimum the following parameters must be set according to your system environment:

- BrokerId
- Class
- ServerName
- Service

For more information see *Configuring the RPC Server*.

## Binding the RPC Executables

The RPC executables are distributed prelinked with unresolved external references to z/OS components. To resolve these external references, you need to relink them in your environment. Execute step “Link the RPC server modules” of job EXPINSTB located in the EXX109.JOBS data set.

## Using z/OS Privileged Services

Some of the RPC server features such as impersonation require privileged z/OS access. Therefore, the RPC server should be started (initially) from an APF-authorized library. Consequently, all other load data sets concatenated to STEPLIB DD have to be APF-authorized as well, including customer's server data sets.

To cope with non-APF-authorized data sets, a server invocation module `EXXAUTH$` is provided. The module can be invoked (APF-authorized) from an APF data set (other than STEPLIB), which installs the authorized PC routines necessary before it invokes (unauthorized) the RPC server.

### ➤ To install the server invocation module

- 1 Copy load module `EXXAUTH$` from EntireX load library `EXP109.LB00` to an APF-authorized load library that is linked to the system LNKST concatenation and rename the copy to `EXXAUTH`.



**Note:** A different name is essential if the `EXP109.LB00` is part of the STEPLIB concatenation, because under z/OS search for the module is first done in the STEPLIB libraries prior to LNKST libraries. Thus not changing the name would result in invoking the module from STEPLIB instead of invoking the module from LNKST and the module could not work properly.

- 2 Change the RPC server startup JCL:

```
//BATRPCS EXEC PGM=EXXAUTH,PARM='RPCSRVB CFG=DD:CONFIG'
```



**Note:** `EXXAUTH` expects the first parameter to be the name of the RPC server to be started. Subsequent parameters will be passed to the RPC server directly.

## Installing the Server-side Mapping Container for an RPC Server for Batch (Optional)

You can skip this step if you are a new customer or have never used server-side mapping files (Designer files with extension .svm) from previous EntireX versions.

If you are using server-side mapping files (Designer files with extension .svm) at runtime, you need to set up a server-side mapping container.

Execute the EXPVSVMAL job located in the EXX109.JOBS data set. After updating the SVMFILE variable to a suitable name, the following steps are performed:

- allocation of the required VSAM cluster
- initialization of the cluster with a dummy server mapping (VSAM record)

Insert the server-side mapping container (VSAM file) into your RPC server's JCL under the DD name of ERXSVM. If this cluster is to be shared between more than one RPC server it should be defined with the RLS=NRI attribute:

```
//ERXSVM DD DISP=SHR,DSN=< batch.svm.cluster >
```

Server mapping files with extension .svm are no longer supported at design time by the Designer. You can still use them at runtime in a server-side mapping container. All special COBOL syntax and features supported by server mapping files with extension .svm are also covered by server mapping files with extension .cvm. See *When is a Server Mapping File Required?*

We recommend migrating .svm files to .cvm files. See *Migrating Server Mapping Files* under *Server Mapping Files for COBOL* in the Designer documentation.

See also *Server Mapping Files for COBOL* | *Server-side Mapping Files* | [Job Replacement Parameters](#).

## Using SSL/TLS Connections with the RPC Server for Batch (Optional)

See *Using SSL/TLS with the RPC Server*.

## Installing the RPC Server for IMS

The EntireX RPC Server for IMS allows standard RPC clients to communicate with RPC servers on the operating system z/OS running with IMS in BMP mode. It supports the programming languages COBOL, PL/I and C and can provide IMS-specific PCB pointers for access to IMS databases if needed.

- [Preparing Your Startup JCL](#)
- [Customizing Your Server Configuration](#)
- [Binding the RPC Executables](#)
- [Installing the Server-side Mapping Container for an RPC Server for IMS \(Optional\)](#)

- [Using SSL/TLS Connections with the RPC Server for IMS \(Optional\)](#)

Prerequisites for all EntireX components are described centrally. See *z/OS Prerequisites*.

## Preparing Your Startup JCL

The RPC Server for IMS can run as a started task. The installation medium contains the following sample JCL:

■ EXPSRVI

### ➤ To prepare your startup JCL

- 1 Modify the sample started task EXPSRVI of the EXX109.JOBS data set to suit your installation.
- 2 Modify the CONFIG DD statement to point to your server configuration file.
- 3 Name a valid PSB in the parameter list for IMSBATCH.
- 4 Concatenate the necessary EntireX product data sets to the RPC server STEPLIB.
- 5 Concatenate your server application data set to the RPC server STEPLIB.
- 6 Insert a valid IMS RESLIB data set high-level qualifier.
- 7 Add the EntireX RPC server JCL to your TASKLIB data set.

## Customizing Your Server Configuration

Modify the sample parameter member CONFIG from EXP109.SRCE. The EntireX RPC server is optimized for use in COBOL environments. Nevertheless, as a minimum the following parameters must be set according to your system environment:

- BrokerId
- Class
- ServerName
- Service

For more information see *Configuring the RPC Server*.



## Binding the RPC Executables

The RPC executables are distributed prelinked with unresolved external references to z/OS components. To resolve these external references, you need to relink them in your environment. Execute step “Link the IMS RPC server modules” of job EXPINSTI located in the EXX109.JOBS data set.

## Installing the Server-side Mapping Container for an RPC Server for IMS (Optional)

You can skip this step if you are a new customer or have never used server-side mapping files (Designer files with extension .svm) from previous EntireX versions.

If you are using server-side mapping files (Designer files with extension .svm) at runtime, you need to set up a server-side mapping container.

Execute the EXPVSVM job located in the EXX109.JOBS data set. After updating the SVMFILE variable to a suitable name, the following steps are performed:

- allocation of the required VSAM cluster
- initialization of the cluster with a dummy server mapping (VSAM record)

Insert the server-side mapping container (VSAM file) into your RPC Server for IMS's JCL under the DD name of ERXSVM. If this cluster is to be shared between more than one server, it should be defined with the RLS=NRI attribute:

```
//ERXSVM DD DISP=SHR,DSN=< ims.svm.cluster >
```

Server mapping files with extension .svm are no longer supported at design time by the Designer. You can still use them at runtime in a server-side mapping container. All special COBOL syntax and features supported by server mapping files with extension .svm are also covered by server mapping files with extension .cvm. See *When is a Server Mapping File Required?*

We recommend migrating .svm files to .cvm files. See *Migrating Server Mapping Files* under *Server Mapping Files for COBOL* in the Designer documentation.

See also *Server Mapping Files for COBOL* | *Server-side Mapping Files* | [Job Replacement Parameters](#).

## Using SSL/TLS Connections with the RPC Server for IMS (Optional)

See also *Using SSL/TLS with the RPC Server* and [Job Replacement Parameters](#).

## Installing RPC Examples for z/OS

---

Under z/OS, all example data sets are delivered in the condensed IBM IEBCOPY load format and can be found with their last level qualifier as a member name in the EXP109.EXPL data set. The CICS, Batch and IMS examples are unloaded with separate jobs:

### ➤ To unload the CICS members to their target data sets and install them in CICS

- Unloading the CICS COBOL or CICS PL/I examples and installation into CICS is part of job `EXPINSTA` located in the `EXX109.JOBS` data set. For CICS, COBOL and PL/I examples are mutually exclusive: you can only install either the COBOL examples or the PL/I examples into the same CICS.

Depending on the installation of the selected programming language examples into CICS, the related Generic RPC Services Module (COBOL | PL/I) used by RPC clients is installed too. CICS definitions for COBOL can be found in member `COBDFH`, and for PL/I in member `PLIDFH` of dataset `EXP109.SRCE`.

For more information refer to the job.

### ➤ To unload the Batch members to their target data sets

- Unloading the Batch COBOL PL/I examples is part of job `EXPEXAMP` located in the `EXX109.JOBS` data set. For more information refer to the job.

### ➤ To unload the IMS members to their target data sets

- Unloading the IMS COBOL examples is part of job `EXPINSTI` located in the `EXX109.JOBS` data set. For more information refer to the job.

### ➤ To compile and link the examples

- Before you can use the examples, you need to build them. Each examples data set has members suffixed with `*IGY`. These members contain JCL to compile and link the related examples. For detailed information see *Delivered Examples* for COBOL Wrapper | PL/I Wrapper.



#### Notes:

1. The examples are described under *Delivered Examples* for COBOL Wrapper | PL/I Wrapper.
2. All z/OS COBOL and PL/I examples are also part of the EntireX Development Tools package, which you can install under Linux and Windows. For installation, see *EntireX Development Tools*.

# 7

## EntireX CICS® Socket Listener

---

The EntireX CICS® Socket Listener is a remote connector on z/OS or z/VSE to call CICS mainframe programs.



**Note:** For configuration, refer to *Preparing for CICS Socket Listener* (EntireX Adapter | RPC Server for CICS Socket Listener).

Execute job `EXPINSTL` located in the `EXX109` data set. This will install the CICS Socket Listener with all relevant CSD entries. This job:

- installs the load modules `EXXRFEC`, `EXXRFECX`, `EXXRFECU`, `EXXRFECA` and `EXXRFECs`
- creates the following CICS CSD definitions:
  - program definitions for the load modules `EXXRFEC`, `EXXRFECX`, `EXXRFECU`, `EXXRFECA` and `EXXRFECs`, where definition for `EXXRFECs` needs `EXECKEY(CICS)`
  - transaction `XRFE`, which is the standard transaction name that can be maintained with the RPC Server or Adapter configuration. If you need to change this CSD entry default, you also need to adapt the configuration parameter `CICS transaction ID`. More information:
    - if you are using the EntireX Adapter, see *Connection Parameters for CICS Socket Listener Connections*
    - if you are using the RPC Server for CICS Socket Listener, see *Configuring an RPC Server Instance > CICS* using the Command Central GUI | Command Line

### ➤ To run the CICS programs in a separate user transaction

- 1 Define your user transaction ID in the RPC Server for CICS Socket Listener, for example:

```
cics.sl.user.transaction.id=UTSK
```

For more information see *Configuring the CICS Socket Listener Side* in the RPC Server for CICS Socket Listener documentation.

- 2 Create the CSD for the user transaction ID.
- 3 In this CSD definition, specify EXXRFECU for the PROGRAM attribute. To define, for example, UTSK as user transaction ID, use the following commands:

```
DEFINE TRANSACTION(UTSK) GROUP(...)  
DESCRIPTION(CICS Socket Listener user transaction)  
PROGRAM(EXXRFECU)
```



**Note:** The CICS Socket Listener is also installed along with the RPC Server for CICS, job EXPINSTA located in EXX109 data set. See [Installing the RPC Server for CICS](#).

# 8

## Installing EntireX Security under z/OS

---

■ Installing EntireX Security for Broker Kernel .....	48
■ Setting up EntireX Security for Broker Stubs .....	51

**Notes:**

1. If you are using EntireX Security, and if your application(s) use ACI version 7 or below, you must install EntireX Security for Broker stubs. For ACI version 8 and above, see *Writing Applications using EntireX Security* to ensure that your application(s) will perform as expected when using EntireX Security.
2. Before installing EntireX Security, make sure that all prerequisites for EntireX components have been met. See *z/OS Prerequisites*.

## Installing EntireX Security for Broker Kernel

---

This section describes the steps for installing EntireX Security for Broker kernel under z/OS. The installation procedure has the following steps:

- [Modify Broker Attribute File](#)
- [Define RACF Resource Profiles](#)
- [Perform Client Authorization Checks \(Optional\)](#)
- [Enable Trusted User ID \(Optional\)](#)
- [Build Language-specific Messages \(Optional\)](#)
- [Start \(Restart\) Broker Kernel](#)

### Modify Broker Attribute File

#### ➤ To modify the Broker attribute file

- 1 Insert the following parameter in the section `DEFAULTS=BROKER` of the Broker attribute file:

```
SECURITY=YES
```

- 2 Modify the Security-specific attributes section of the Broker attribute file according to your requirements. These parameters are used to determine whether you will use SAF Security or LDAP-based authentication. See *Security-specific Attributes*. If you are using LDAP-based authentication, authorization checks are not available to you.



**Note:** Setting `SECURITY=YES` will load the provided LOAD module `USRSEC` from the `EXB109.LOAD` library. This module will perform privileged operations, such as execute the `RACROUTE`, requiring APF authorization.

## Define RACF Resource Profiles

EntireX Security performs checks against user profiles and resource profiles represented in RACF, CA ACF2, and CA Top Secret. See *Resource Profiles in EntireX Security*.

## Perform Client Authorization Checks (Optional)

For services supporting Natural RPC or other applications that use RPC, you can perform authorization checks on the client by defining the "per service" attribute `CLIENT-RPC-AUTHORIZATION=YES` in the Broker attribute file. Setting this parameter to `YES` will cause the RPC library and program names to be appended to the profile associated with the authorization check. The resource profile would then appear as follows:

```
Class.server.service.rpc-library.rpc-program
```

If the total length of the resource profile exceeds 80 bytes, increase the parameter `MAX-SAF-PROF-LENGTH`.

This check applies only to the client and not the server. `CLIENT-RPC-AUTHORIZATION=YES` should not be set for any services which do not utilize RPC protocol.



**Note:** Natural Security performs its resource authorization checks as follows:

```
<prefix-character>.rpc-library.rpc-program
```

To allow conformity with Natural Security, the `CLIENT-RPC-AUTHORIZATION` parameter can optionally be defined with a prefix character as follows:

```
CLIENT-RPC-AUTHORIZATION=(YES,<prefix-character>).
```

## Enable Trusted User ID (Optional)

If you use the trusted user ID option, set the parameter `TRUSTED-USERID=YES` in the `DEFAULTS=SECURITY` section of the attribute file.

- The trusted user ID feature automatically acquires the identity of the logged-on user or batch job. It must therefore only be used with TP monitors running under the control of RACF, CA ACF2 or CA Top Secret. Batch jobs must run under an identifiable user ID, as inherited by the job submitter, scheduler, or other means.
- Applications using the trusted user ID feature must execute under z/OS and on the same machine, or another z/OS machine connected to Broker through Entire Net-Work. Communication is through the Adabas SVC mechanism.
- Applications must not provide a password if they intend to use trusted user ID. This applies to the EntireX RPC Server for CICS | Batch | IMS, and also the Natural RPC Server. If the application cannot avoid supplying a password, it is permissible to assign a password value of `NOPASSWORD`.
- EntireX Security trusted user ID functionality is relevant only for determining the z/OS user ID associated with applications executing on z/OS which communicate with EntireX Broker, which are also executing on z/OS via the Adabas SVC mechanism. It cannot be used in configurations

which include application components executing on separate, non-z/OS computers that communicate with EntireX Broker through Entire Net-Work. Such configurations invalidate the usage of trusted user ID.

- The SVCSAF module is supplied with EntireX. If your Adabas version is lower than 8.2, the resulting Adabas SVC must be linked into an APF-authorized library. Since Adabas 8.2, the SAF component is linked to ADASVC by default. Linkage example:

```

/*-----
/* CREATE A NEW ADASVC MODULE THAT INCLUDES SVCSAF MODULE
/*-----
//LNKSVC EXEC PGM=IEWL,PARM='XREF,LIST,LET,NCAL,RENT,REUS'
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD SPACE=(CYL,(1,1)),UNIT=VIO
//WALLIB DD DISP=SHR,DSN=WAL842.LOAD ADASVC and SVCSAF
//SYSLMOD DD DISP=SHR,DSN=WAL842.NEW.LOAD NEW ADASVC OUTPUT
//SYSLIN DD *
MODE AMODE(31),RMODE(24)
INCLUDE WALLIB(ADASVC)
SETCODE AC(1)
INCLUDE WALLIB(SVCSAF)
NAME ADASVC(R)
/*
//*

```

- Implementing the SAF trusted user ID option in EntireX Security under CICS TS version 1.2 and above requires the installation of the Adabas task related user exit (ADATRUE) and setting either the ADAGSET or LGBLSET (depending upon the Adabas version) parameter SAF=YES. See *Installing Adabas with TP Monitors* in the Adabas installation documentation for complete details on installing ADATRUE. Samples of the ADAGSET and LGBLSET parameter modules can be found in the library WAL842.SRCE.

For additional supporting information, see the *Installation Procedure* section of the *Adabas Installation Manual*.

## Build Language-specific Messages (Optional)

### ➤ To build language-specific messages

- 1 Copy the template message module EXX109.SRCE(NA2MSG0) to another member - for example, EXX109.SRCE(NA2MSG9) - and then modify the message texts to suit your own language requirements.



**Note:** NA2MSG0, NA2MSG1, and NA2MSG2 are reserved names.



- 2 Assemble and link your modified source module using the sample JCL EXX109.SRCE(SAGJ106), ensuring that you create a unique load module in the EXX109.LOAD library.
- 3 Modify the `ERRTXT-MODULE` parameter in the `DEFAULTS=SECURITY` section of the attribute file to reflect the name of your unique load module.

### Start (Restart) Broker Kernel

The Broker must be restarted to pick up changes to the Broker attribute file and to initialize Broker kernel under z/OS to perform security checks.

Basic installation of EntireX Security for Broker kernel is now complete.

## Setting up EntireX Security for Broker Stubs

This section describes the steps for installing EntireX Security for Broker stub under z/OS. The installation consists of the following steps:

- [Assemble the SAFCFG Configuration Module](#)
- [Link the Security Components](#)
- [Rename SECUEXIT](#)



**Note:** If you are running your application(s) at ACI version 7 or below, the following steps are required to install EntireX Security for the Broker stubs in all environments where applications execute either as clients or servers. See *Platform Coverage* to see where EntireX Security for broker stubs is supported. These steps are not required if you are running your application(s) at ACI version 8 or above.

### Assemble the SAFCFG Configuration Module

The SAFCFG configuration module is required for applications running on z/OS using ACI version 7 or below.

#### ➤ To assemble the SAFCFG configuration module

- Run job `WALvrs.JOBS(SAFI010)`, which assembles and links SAFCFG (load module).



**Note:** This module comes with preconfigured defaults. See source module `WALvrs.SRCE(SAFCFG)`.

## Link the Security Components

For applications running on z/OS using ACI 7 or below, the Broker stub security component must be linked with the following stubs: BROKER, CICSETB, NATETB23, COMETB, MPPETB.

### ➤ To link the Broker stub security component

- Relink all applications that contain ACI stub modules BROKER, CICSETB, NATETB23, COMETB, or MPPETB to include the following modules:

- NA2PETS Broker security stub logic module
- SAFCFG System parameter module

See *Administering Broker Stubs*.

Location of sample INCLUDE statements: EXX109.JOBS(EXXJ109).



**Note:** These components are needed for backward compatibility if your applications issue any commands using ACI version 7 or below. Applications using ACI version 8 or above do not require these additional components in the stubs. For ACI version 7 or below, these components must be added to the stub environment utilized by the application. Failure to link these components along with the stub when using ACI version 1 through 7 can result in message "SEFM225 MESSAGE FROM BACK LEVEL STUB" being issued by Broker kernel.

## Rename SECUEXIT

SECUEXIT must be made available for applications running on z/OS using ACI version 7 or below.

### ➤ To make SECUEXIT available

- 1 Rename SECUEXIT to SECUEXIT in library EXX109.LOAD so that it is available to applications running the IBM C stub.
- 2 Ensure that SECUEXIT is available in EXX109.LOAD for all applications.



#### Notes:

1. These steps are needed for backward compatibility if your applications issue any commands using ACI version 7 or below. Applications using ACI version 8 or above do not require these additional components in the stubs.
2. For ACI version 7 or below, these components must be added to the stub environment utilized by the application.

Installation of EntireX Security for Broker stubs is now complete. Now you can install the security components for the Broker stubs on the remaining operating systems where your application components are located.

See also *Setting up EntireX Security for Broker Stubs* under Linux | Windows.



# 9

## Installing EntireX Java Components under z/OS UNIX

---

■ Scope .....	56
■ Installation Steps .....	56
■ Configuring and Administering the Java Components .....	57

This chapter describes how to install EntireX Java components under z/OS UNIX.

## Scope

---

The following EntireX Java components can be used under z/OS UNIX. The links provide more information on configuring and administering these components under Linux.

- *RPC Server for Java*
- *RPC Server for XML/SOAP*
- *Listener for XML/SOAP*
- *RPC Server for IBM MQ*
- *Listener for IBM MQ*
- *RPC-ACI Bridge*
- *RPC Server for CICS IPIC*
- *RPC Server for CICS ECI*
- *RPC Server for CICS Socket Listener*
- *RPC Server for IMS Connect*

## Installation Steps

---

### » To install EntireX Java components under z/OS UNIX

- 1 Unload the delivered installation medium to your DASD device. See [Copying the Contents of the Installation Medium to Disk](#).
- 2 On your UNIX system, create a directory for the installation. For example:

```
mkdir SoftwareAG; cd SoftwareAG;
```

- 3 Copy the file *hlq.EXXvrs.TAR* to the created directory, where *hlq* is your high-level qualifier and *vrs* is the product version, release and service pack:

```
tso "oput 'hlq.EXXvrs.TAR' 'SoftwareAG/exxvrs.tar' binary"
```

#### 4 Unpack the TAR file:

```
tar xvfo exxvrs.tar
```

Now you have created a folder structure with subdirectories EntireX and WS-Stack which contain all necessary files in several subdirectories. In the `EntireX/bin` directory you have an environment file `exx_zos_env.sh`, which you can source to set your environment. Before sourcing the environment, edit this file and set the environment variable `EXXDIR` to your EntireX directory. Then you can enter the following command (note the space after the initial period):

```
. ./exx_zos_env.sh
```

Check your Java installation and edit the file `exxjenv.sh` in the `EntireX/bin` directory. You need to set the `JAVA_HOME` environment variable to your corresponding Java installation.

Bourne shell scripts are provided in the `EntireX/bin` directory to start the different RPC servers.

To use the RPC Server for XML/SOAP, configure the path to the `example.xmm` file. Edit the file `EntireX/config/entirex.xmlrpcserver.configuration.xml` and adapt the pathname in the `exx-xmm` tag. This file is already in EBCDIC format and the encoding has been set to CP037. If you need a different encoding, change this setting accordingly.

## Configuring and Administering the Java Components

The following table shows the provided start script and configuration file for each RPC server or listener.

RPC Server/Listener	Script in EntireX/bin	Config File in EntireX/config
RPC Server for Java	<code>jrpcserver.bsh</code>	<code>entirex.javarpserver.properties</code>
RPC Server for XML/SOAP	<code>jxmlrpcserver.bsh</code>	<code>entirex.xmlrpcserver.properties</code>
RPC Server for IBM MQ	<code>wmqbridge.bsh</code>	<code>entirex.wmqbridge.properties</code>
Listener for IBM MQ	<code>wmqlistener.bsh</code>	<code>entirex.wmqbridgelistener.properties</code>
RPC-ACI Bridge	<code>jaci-rpc.bsh</code>	<code>entirex.aci-rpc.properties</code>
RPC Server for CICS IPIC	<code>cicsipicserver.bsh</code>	<code>entirex.cicsipic.properties</code>
RPC Server for CICS ECI	<code>cicseciserver.bsh</code>	<code>entirex.cicseci.properties</code>

RPC Server/Listener	Script in EntireX/bin	Config File in EntireX/config
RPC Server for CICS Socket Listener	cicsocketlistenerserver.bsh	entirex.cicsocketlistener.properties
RPC Server for IMS Connect	imsconnectserver.bsh	entirex.imsconnect.properties

For the Listener for XML/SOAP you need to set up a Web server and install Software AG Web Services Stack to this Web server. See *Configure the Web Services Stack Runtime* in the *Software AG Infrastructure Administrator's Guide* under Software AG Suite & Cross-Product Guides. After successful installation, copy or upload `<inst_dir>/EntireX/classes/entirex.jar` to the WEB-INF/lib folder in the Web Services Stack application folder.



# 10

## Installing the EntireX ICU Custom Converter Build Environment under z/OS UNIX

---

■ Installation Steps .....	60
----------------------------	----

This chapter describes how to install the EntireX ICU Custom Converter Build Environment under z/OS UNIX. These steps are necessary if you want to use user-written ICU custom converters. See *Building and Installing ICU Custom Converters* in the z/OS Administration documentation.

## Installation Steps

---

### ➤ To install the ICU Custom Converter Build Environment

- 1 Unload the delivered installation medium to your DASD device. See [Copying the Contents of the Installation Medium to Disk](#).
- 2 On your UNIX system, create a directory for the installation. For example:

```
mkdir CustomConverter; cd CustomConverter
```

- 3 Copy the file *hlq.EXXvrs.TICU* to the created directory,

where *hlq* is your high-level qualifier and

*vrs* is the product version, release and service pack:

```
tso "oput 'hlq.EXXvrs.TICU' 'exxvrs.icu.tar' binary"
```

- 4 Unpack the TAR file:

```
tar xvf exxvrs.icu.tar
```

Now you have extracted the `makeconv` utility and ICU shared libraries. These binaries are needed to execute `makeconv`.

# 11

## Verifying the z/OS Installation

---

■ EntireX Broker .....	62
■ Sample Programs for Client (BCOC) and Server (BCOS) .....	62
■ RPC Server for CICS .....	63
■ RPC Server for Batch .....	63
■ RPC Server for IMS .....	63

## EntireX Broker

---

### > To test the EntireX Broker installation

- 1 Start EntireX Broker. See *Starting and Stopping the Broker* in the z/OS Administration documentation.
- 2 Run the example programs:
  - BCOS is the example server program
  - BCOC is the example client program

See [Sample Programs for Client \(BCOC\) and Server \(BCOS\)](#).

## Sample Programs for Client (BCOC) and Server (BCOS)

---

The programs BCOC and BCOS are client and server programs provided for test purposes. They are delivered as load modules in EXX109.LOAD.

The execution JCL for the example programs is located in the EXX109.JOBS library in members EXXBCOS and EXXBCOC. Modify the JCL for your installation, and submit job EXXBCOS first, followed by job EXXBCOC.

A single message will be sent by program BCOC and then received by program BCOS. Both jobs should return a zero condition code. The output from each job is written to SYSPRINT.

### BCOC Parameters

install\_verifyClientServer\_bcoc

### BCOS Parameters

install\_verifyClientServer\_bcos

## RPC Server for CICS

---

The installation can be verified by running an example program provided for z/OS. The delivered COBOL and PL/I examples need to be installed separately (either COBOL or PL/I examples). See [Installing RPC Examples for z/OS](#).

See also:

- *Client and Server Examples for z/OS CICS (COBOL | PL/I)* in the respective Wrapper documentation
- *Starting the RPC Server* in the RPC Server for CICS documentation

## RPC Server for Batch

---

The installation can be verified by running an example program provided for z/OS. The delivered COBOL and PL/I examples need to be installed separately (either COBOL or PL/I examples or both). See [Installing RPC Examples for z/OS](#).

See also:

- *Client and Server Examples for z/OS Batch (COBOL | PL/I)* in the respective Wrapper documentation
- *Starting the RPC Server* in the RPC Server for Batch documentation

## RPC Server for IMS

---

The installation can be verified by running an example program provided for z/OS. The delivered COBOL and PL/I examples need to be installed separately (either COBOL or PL/I examples or both). See [Installing RPC Examples for z/OS](#).

See also:

- *Client and Server Examples for z/OS IMS BMP (COBOL | PL/I)* in the respective Wrapper documentation
- *Starting the EntireX RPC Server (IMS)* in the RPC Server for IMS documentation



# 12

## Installing Adabas Components for EntireX under z/OS

---

■ Initializing the Adabas Communication Environment .....	66
■ SVC Integrity Validation .....	76
■ Requirements for Cross-Memory Services .....	77
■ Applying Zaps .....	78

## Initializing the Adabas Communication Environment

---

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

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

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

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

ADASVC installation can be either temporary or permanent:

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



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



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

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

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

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



- [Using ADASIR](#)

## SVC Compatibility Issues Between Adabas Releases

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

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

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

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

## Authorization Requirements

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



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

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

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

```
ADABAS.SVCsss.IDdddd
```

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

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

```
ADABAS.SVCsss.IDdddd.Nucnnnnn
```

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

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

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

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

**Note:**

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

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

### Allocating an SVC Table Entry

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

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

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

### Subsystem Name Requirements

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

### Page-Fixing the Adabas SVC

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

### Initializing the Adabas SVC

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

### Router Installation Overview

- [Temporary Router Installation \(SMA Job Number I011\)](#)
- [Permanent Router Installation \(SMA Job Number I010\)](#)

#### Temporary Router Installation (SMA Job Number I011)

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

#### ➤ to perform temporary router installation:

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

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

**Permanent Router Installation (SMA Job Number I010)**

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

➤ **to perform permanent router installation:**

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

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

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

- 3 If you wish to associate a subsystem name other than the default ADAB with ADASVC, customize sample job JCL SVC step SUBSYS to invoke IBM utility IMASPZAP to set the desired 4-character subsystem name.
- 4 Add a subsystem entry to SYS1.PARMLIB member IEFSSN<sub>xx</sub> for the subsystem name associated with the SVC. Customize and run the job JCLUPDT to add a new subsystem definition entry in the correct format.

Or:

Make the update with an on-line editor.

- 5 Customize sample job JCLSIR to copy ADASIR to SYS1.LINKLIB or a library concatenated with it by SYS1.PARMLIB member LKNLST<sub>xx</sub>.



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

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

## Using ADASIP for Temporary Installations

- [ADASIP Functions](#)
- [ADASIP Parameters](#)
- [Executing ADASIP](#)

### ADASIP Functions

ADASIP performs the following functions:

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

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

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

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

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

## ADASIP Parameters

ADASIP parameters have the following syntax:

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

— where

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

The following are valid ADASIP parameter abbreviations:

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

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

are passed to ADASIR without being verified.

### Executing ADASIP

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

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

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

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

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

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

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

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

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

Note these restrictions:

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

```

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

```

—results in an equivalent line of

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

## Using ADASIR

- [ADASIR Functions](#)
- [ADASIR Parameters](#)
- [Executing ADASIR](#)

### ADASIR Functions

The ADASIR program is invoked

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

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

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

### ADASIR Parameters

ADASIR parameters have the following syntax:

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

Variable	Description
<i>i</i>	The ID table subpool: 228 for fixed CSA or 241 (the default) for pageable CSA.
<i>l</i>	Indicates whether message ADAS11 or ADAS12 is to be displayed on the operator console: Y for yes or N (the default) for no.
<i>n</i>	The ID table entry count, which can range from 1 to a maximum specified at offset X'146' in the CSECT IEAVESVT of the z/OS nucleus (see section <a href="#">Requirements for Cross-Memory Services</a> ).
<i>svcn</i>	The Adabas SVC number (200-255).



Variable	Description
<i>svct</i>	Indicates whether or not the operator should be prompted for permission to update the SVC table entry. Enter P (the default) to receive a prompt, or N for no prompt. P is recommended if a possibility exists that the SVC table entry will not be what ADASIR expects.
<i>class</i>	is the System Authorizatin Facility (SAF) class name used to test authorization to start a target (default FACILITY).

The following are valid abbreviations for ADASIR parameters:

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

### Executing ADASIR

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



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

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

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

—where

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

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

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

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

## SVC Integrity Validation

---

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

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

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

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

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

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

## Requirements for Cross-Memory Services

---

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

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

The ASID representing this address space is not reassigned until the next IPL. Therefore, you should choose a sufficiently high value for the `MAXUSERS` parameter in the active `IEASYSxx` member of `SYS1.PARMLIB` or - if your system supports it - the `RSVNONR` parameter in the same member can be adjusted accordingly. Also, the Adabas nucleus should not be stopped and started without good reason.

This is described in the manuals referred to in the topics Recovery Considerations and Resource Management. Additional information can be found in IBM APARs OZ61154, OZ61741, and OZ67637.

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

If your configuration is using z/OS 1.6 or later and your hardware supports the Address Space and LX Reuse Facility (ALRF), the version 8 ADASVC will make use of reusable system LXs. Otherwise, a non-reusable system LX will be used as in previous releases. Reusable system LXs resolve the constraints imposed by non-reusable LXs. The remainder of this section discusses these constraints.

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

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

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

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

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

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

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

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

## Applying Zaps

---

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

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

—where the following are examples of zap control statements:

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



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



# 13

## Installing Adabas with TP Monitors

---

■ Preparing Adabas Link Routines for IBM Platforms .....	82
■ Installing Adabas with IMS TM under Adabas 8 .....	84
■ General Considerations for Installing Adabas with CICS .....	86
■ Installing Adabas with CICS under Adabas 8 .....	88
■ Installing Adabas with Com-plete under Adabas 8 .....	103
■ General Considerations for Installing Adabas with Batch/TSO .....	105
■ Installing Adabas with Batch/TSO under Adabas 8 .....	107
■ Modifying Source Member Defaults (LGBLSET Macro) in Version 8 .....	109

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

## Preparing Adabas Link Routines for IBM Platforms

---

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

- [Addressing Mode Assembly Directives in Adabas Link Routines](#)



**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.

### Addressing Mode Assembly Directives in Adabas Link Routines

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

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

### Re-linking Adabas 8 Link Routines

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

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

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





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

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

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

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

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

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

### **ADAUSER AMODE/RMODE Considerations**

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.

## Installing Adabas with IMS TM under Adabas 8

---

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

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

### Hint:

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

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



**Note:** If CALL 'ADABAS' results in a dynamic load of the Adabas link routine, the load module name of the Adabas link routine must be called ADABAS. As ADALNI saves the address of its working storage in itself, ADALNI must never be unloaded in the MPP. Otherwise you will get memory leaks.

This section covers the following topics:

- [IMS TM Link Routines for Adabas 8](#)
- [Obtaining the Adabas User ID](#)
- [Obtaining the SAF ID](#)

## ■ Installation Procedure under Adabas 8

### IMS TM Link Routines for Adabas 8

These are Adabas 8 link routines for IMS TM:

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

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

The Adabas Version 8 ADALNI and ADALNK are UES-enabled as distributed.

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

### Obtaining the Adabas User ID

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

### Obtaining the SAF ID

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

### Installation Procedure under Adabas 8

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

- 1 Copy the sample member LNIGBL provided in the Adabas 8 AII<sub>vrs</sub>.SRCE library to any appropriate user source library where they can be modified. These modules contain LGBLSET parameters that are used to create default settings for link components. A complete description of LGBLSET parameters can be found in [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), elsewhere in this guide.
- 2 Modify the LNIGBL member in the user source library.



**Note:** The OPSYS parameter must be set to ZOS.

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

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

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

- 4 Copy sample job LNKLN18 to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the ADALN18 base module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from this job to an appropriate user load library. LNKLN18 can be found in the Adabas 8 AII*vrs*.SRCE library.



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

The module resulting from this job is ADALNI.

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

The Adabas 8 link routine is prepared.

## General Considerations for Installing Adabas with CICS

---

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

The Adabas command-level link routine supports the CICS transaction server (CTS) environment. The CICS components from Adabas 8 are required when running with an Adabas 8 SVC.

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

- [Adabas Bridge for VSAM Considerations](#)
- [CICS MRO Environment Requirements](#)
- [Using CICS Storage Protection](#)

- [Sample Resource Definitions](#)
- [Requirement for CICS Command Resource Security](#)

## Adabas Bridge for VSAM Considerations

If you are running Adabas Bridge for VSAM 4.2 or 5.1 under CICS, you must run CICS 3.3 or above and the Adabas Version 8 or above command-level link routine.

## CICS MRO Environment Requirements

If you run the Adabas CICS command-level link routine with the CICS multiple region option (MRO), you must set the `MRO` parameter to "YES" and use the default for the `NETOPT` parameter. In an Adabas 8 installation, these parameters are supplied via the `LGBLSET` macro (read [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), elsewhere in this section).

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

Alternatively, you can create a user exit for the link routine that:

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

This exit is link user exit 1 (`LUEXIT1`). The exit then allows the Adabas SVC to provide a proper Adabas communication ID in the Adabas command queue element (CQE) even when transactions originate in multiple regions.

## Using CICS Storage Protection

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

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

Transaction isolation is an extension of the storage protection mechanism. It further protects CICS resources by isolating them in subspaces. This protects user key resources from one another, and protects CICS key resources from the CICS kernel. Transaction isolation can be enabled globally through the CICS `TRANISO` system initialization (`SIT`) parameter, and for each CICS transaction with the new resource definition `ISOLATE` keyword. Transaction isolation places some restrictions on CICS resources that must be available both during the life of the CICS system and to all transactions running in the CICS system.

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

### Sample Resource Definitions

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 CICS command-level source library (ACI<sub>vr</sub>s.SRCE).

### Requirement for CICS Command Resource Security

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

## Installing Adabas with CICS under Adabas 8

---

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

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

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

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

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

- the **Adabas CICS stub**, ADACICS

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

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

This section covers the following topics:

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

## The Adabas CICS Application Stub (ADACICS)

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

## The Adabas CICS Names Module (ACINAMES)

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

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



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

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

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

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

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

### The MACINS Macro

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

The MACINS macro is provided in the Adabas CICS z/OS source library. A sample ACINAMES source member is provided in the ACI<sub>vr</sub>s source library on z/OS systems.

The syntax of the MACINS macro is shown below:

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

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



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

### Example

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

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

### The MACIOPT Macro

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

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

The syntax of the MACINS macro is shown below:

```

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

GROUP , GTNAME = link-globals-table-name

FINAL

```

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

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

### The ENTRY=GLOBAL Statement

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

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

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

Parameter	Description	Default
	We recommend that you use the default load module name of ACIOPT, defining ACIOPT to CICS and allowing ADACIC0 to load the ACIOPT module when the program is executed to install the Adabas CICS components.	
IMSGDEST	<p>Identifies the destination type for the installation progress and error messages produced by ADACIC0: console, transient data queue, or both.</p> <p>IMSGDEST=CONSOLE is the default and causes all installation messages to be written to the console with EXEC CICS WRITE OPERATOR commands. This is how messages for previous Adabas CICS components produced installation messages.</p> <p>IMSGDEST=TDQ causes ADACIC0 to determine if a named CICS transient data queue is available and, if so, to write installation progress and error messages to that queue. If IMSGDEST=TDQ is specified, the IMQNAME parameter must also be specified to provide the name of the CICS transient data queue for the messages. If the named transient data queue is not enabled and open, messages will be written to the console. No error message is written to indicate that the transient data queue could not be used. If the CICS transient data queue is open and enabled, message ADAK001 is written to the console to indicate that all further messages will be written to the CICS transient data queue. If, during ADACIC0 processing, the transient data queue becomes unavailable, subsequent messages will be written to the console.</p> <p>IMSGDEST=BOTH causes installation progress messages to be written both to the console and to a named CICS transient data queue.</p>	CONSOLE
IMQNAME	<p>Specifies the 4-character name of the CICS transient data queue where installation progress and error messages should be written. If IMQNAME is specified then the IMSGDEST parameter must be set to TDQ or BOTH.</p> <p>The named transient data queue must be defined to CICS as either an extra-partition queue or as an indirect queue which references an extra-partition data queue. The simplest way to set up such a data queue is to make it indirect and refer to the CICS-supplied extra-partition data queue CSSL.</p> <p>The queue may be defined using the CICS RDO facility (using the CEDA transaction) or using the DFHDCT macro. For more information, consult the appropriate IBM CICS documentation.</p> <p>Installation messages written to a CICS transient data queue are variable length records with no printer control character in the first byte of the record. The records will not exceed 132 bytes in length.</p>	There is no default.
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</p>	8

Parameter	Description	Default
	to limit the storage and resource constraints multiple Adabas CICS execution units might place on your system. Although the setting for MNTRUE may be quite high, the storage, resources and Adabas CICS components must be available to be installed.	

### The ENTRY=GROUP Statement

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

Only one parameter can be specified for ENTRY=GROUP:

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

### The ENTRY=FINAL Statement

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

There are no parameters for the ENTRY=FINAL statement

### Example

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

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

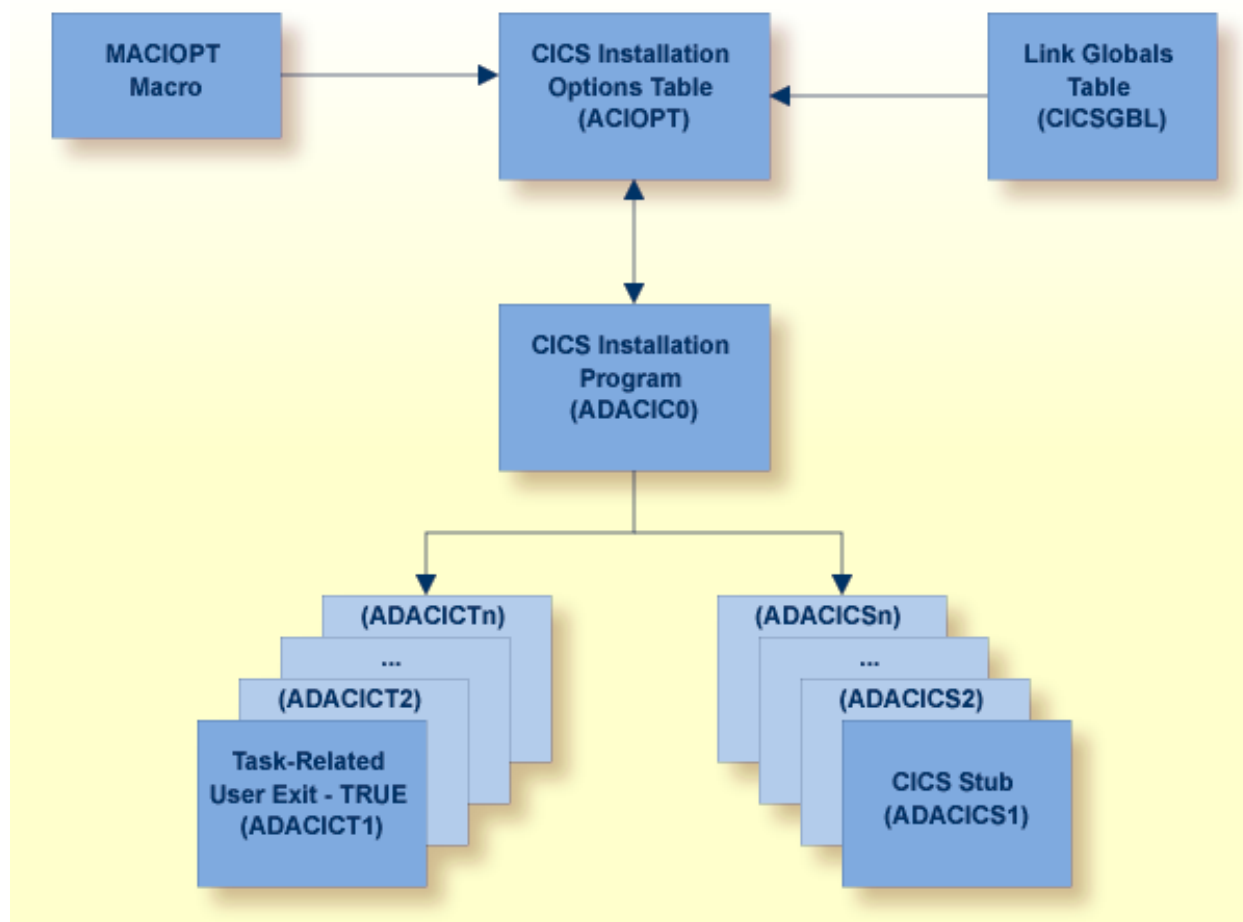
## **Adabas Task-Related User Exits (TRUEs)**

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

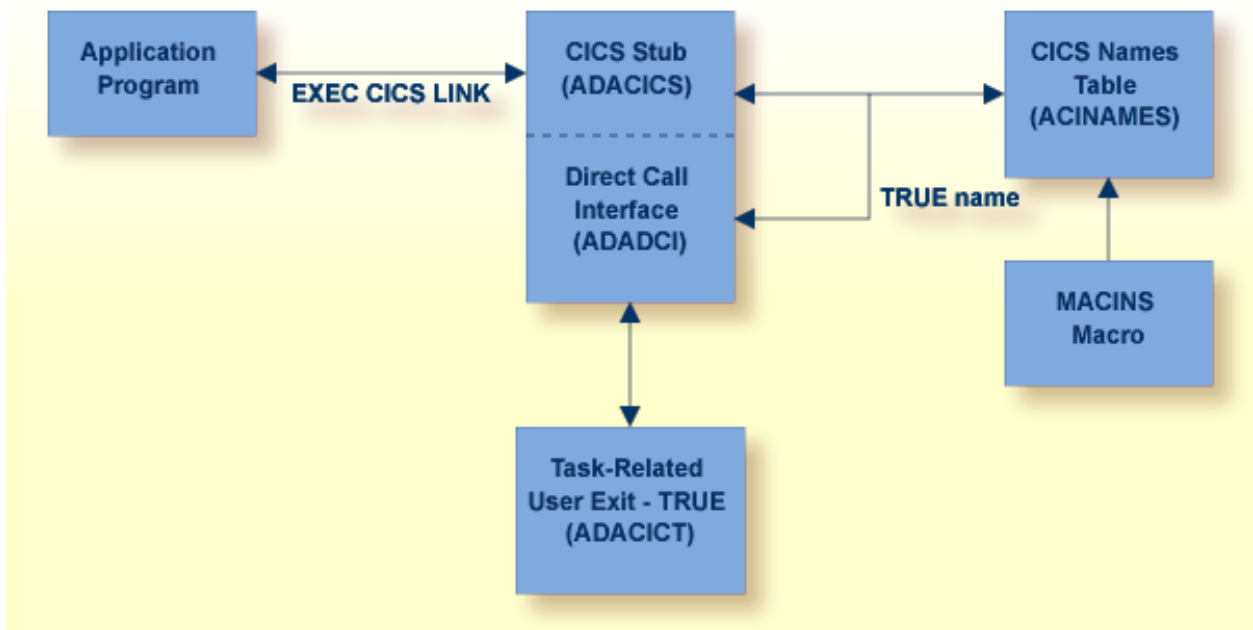
If the Adabas CICS interface employs exits such as the Adabas Fastpath exit or other System Coordinator facilities, there may be more than one invocation of the Adabas TRUE for each EXEC CICS LINK issued by the application program. Other Software AG products that can have multiple TRUE invocations for 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 with CICS under Adabas 8.

Module	Description
ADACIC0	CICS installation program
ADACICS	CICS command-level module.
ADACICT	CICS task-related user exit (TRUE) module.
ASMCINS	Assembles the ACINAMES module and links it with the Adabas CICS application stub
ASMCOPT	Assembles and links the ACIOPT source module as the standalone module named "ACIOPT"

## Installation Procedure

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

- Step 1. Copy the Load Modules
- Step 2. Prepare the Adabas CICS Installation Options Table
- Step 3. Prepare the Adabas CICS Task-Related User Exits (TRUEs) -- ADACICT
- Step 4. Prepare the Adabas CICS Names Module -- ACINAMES
- Step 5. Prepare the Adabas CICS Application Stub -- ADACICS
- Step 6. Prepare the CICS Link Globals Table -- CICSGBL
- Step 7. Assemble the CICS Link Globals Table -- ASMGBLS
- Step 8. Link the Assembled CICS Link Globals Table -- LKNGCICS
- Step 9. Modify CICS Installation Values -- DEFADAC

- [Step 10. Update the CICS CSD File](#)
- [Step 11. Modify, Assemble and Link the CICS PLTPI Table for ADACIC0](#)
- [Step 12. Start the CICS](#)

### Step 1. Copy the Load Modules

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

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

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

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

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

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

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

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

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

- The MACIOPT ENTRY=GLOBAL specification must be first specification in the source member; only one MACIOPT ENTRY=GLOBAL specification can be made per ACIOPT generation.
- The MACIOPT ENTRY=FINAL specification must be the last entry for the ACIOPT generation; only one MACIOPT ENTRY=FINAL specification can be made per ACIOPT generation.



- Multiple MACIOPT ENTRY=GROUP entries may be specified, but they must follow the MACIOPT ENTRY=GLOBAL specification and precede the MACIOPT ENTRY=FINAL specification in the source member.

The MACIOPT macro is located in the ACI<sub>vrs</sub> source library on z/OS systems. For complete information on the MACIOPT macro, read [The MACIOPT Macro](#), elsewhere in this section. A sample ACIOPT source member is provided in the ACI<sub>vrs</sub> source library on z/OS systems.

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

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

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

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



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

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

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

A sample job, LNKATRU, is provided in the ACI<sub>vrs</sub> source library. This sample links the Adabas TRUE with a load module named ADATRU so that it can be installed and referenced in CICS.

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

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

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

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

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

Sample job, ASMCINS, which is provided in the ACI<sub>vr</sub>s source library, assembles the ACINAMES module and links it with the Adabas CICS application stub (see [Step 5](#)), and names the stub "ADABAS".

### Example

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

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

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

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

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

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

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

Sample job, ASMCINS, which is provided in the ACI<sub>vr</sub>s source library, assembles the ACINAMES module and links it with the Adabas CICS application stub (see [Step 4](#)), and names the stub "ADABAS".

### Example

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

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

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

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

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

Modify the sample CICSGBL member found in the Adabas 8 ACI<sub>vr</sub>s.SRCE library. This member contains sample default installation (LGBLSET) parameter settings. For more information about what to modify in this member, read [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), 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 LGBLSET macro is located in the Adabas source library.

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

- 1 Code the link globals table using the LGBLSET macro as described in [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), elsewhere in this section.
  - The OPSYS parameter must be set to ZOS.
  - Be sure to code the ENTPT and TRUENM parameters on each LGBLSET macro so they match the intended Adabas CICS stub name and Adabas CICS TRUE name to be used in a given Adabas CICS execution unit. The Adabas CICS installation program attempts to load each globals table in turn and uses the loaded table to provide the data required to install and activate the components of the execution unit.
  - In CICS environments only, if you want your security system user IDs to be stored in Adabas user queue elements (making them available for display and review as well as preventing response code 200, ADARSP200, subcode 21 when ADARUN SECUID=REQUIRE

is in effect for Adabas), you must code the SAF parameter as YES. This is only required in CICS environments; in other environments, the security system user IDs are automatically stored.

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

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

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

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

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

### **Step 9. Modify CICS Installation Values -- DEFADAC**

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

### **Step 10. Update the CICS CSD File**

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

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

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

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

## Step 12. Start the CICS

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

## Installing Adabas with Com-plete under Adabas 8

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

Software AG's TP monitor, Com-plete requires an Adabas link routine if it is to communicate with Adabas databases, use Software AG's Entire Net-Work product, or use products like Entire System Server running under Com-plete. Com-plete must be run with an Adabas 8 link routine.

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

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

Module	Description
ADALCO8	Base module
ADALCO	Executable default module

### » To prepare the Adabas 8 link routine:

- 1 Copy sample member LCOGBL provided in the Adabas 8 ADA<sub>vrs</sub>.SRCE library to any appropriate user source library where it can be modified (where *vrs* is the number of the latest Adabas version delivered on the tape). LCOGBL is a module containing LGBLSET parameters that are used to create default settings for command-level link components. A complete description of LGBLSET parameters can be found in [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), elsewhere in this guide.
- 2 Modify the LCOGBL member in the user source library.

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

Parameter	Specify...
LOGID	<p>The default database or target ID. This should be a numeric value between "1" and "65535". The default value is "1".</p> <p><b>Note:</b> Specifying default values for LOGID and SVCNO under Com-plete is for documentation purposes only. The ADASVC Com-plete runtime control statements will provide the supported database ID/Adabas SVC combinations to be used in running Com-plete. For more information, read the Com-plete documentation.</p>
SVCNO	<p>The default Adabas SVC number. For z/OS, this number should be between "200" and "255".</p> <p><b>Note:</b> Specifying default values for LOGID and SVCNO under Com-plete is for documentation purposes only. The ADASVC Com-plete runtime control statements will provide the supported database ID/Adabas SVC combinations to be used in running Com-plete. For more information, read the Com-plete documentation.</p>
OPSYS	<p>The three-character abbreviation for the operating system under which Com-plete executes. Valid values include "ZOS" and "VSE".</p> <p><b>Note:</b> The OPSYS parameter must be set to ZOS.</p>
TPMON	COM. This keyword specifies the three-character TP monitor abbreviation. For Com-plete, this value should be "COM".
RENT	YES. This keyword indicates whether or not the module is serially reentrant. For Com-plete, this value should be "YES".
GEN	CSECT. This keyword indicates whether a CSECT or DSECT is generated. CSECT must be specified so an object module is generated that can be linked as the link routine globals load module.
UES	Whether Adabas Universal Encoding Support (UES) should be enabled. The default is YES. .
exit parameters	Whether any other exits are to be active, and in the case of user exits you provide, specify the user exit module names. Specify this information in other parameters of LGBLCOM, as described in <a href="#">Modifying Source Member Defaults (LGBLSET Macro) in Version 8</a> , elsewhere in this guide.

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

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

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

- 4 Copy sample job LNKLCO8 to a user source library and modify it to link the new link globals module you created in the previous step and any required exits with the ADALCO8 base

module. Instructions for modifying the sample job are described at the top of the job. Be sure to direct the output from this job to an appropriate user load library. LNKLCO8 can be found in the Adabas 8 ADA<sub>vr</sub>s.JOBS library.

The module resulting from this job is called ADALCO.

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

The Adabas 8 link routine is prepared.

## General Considerations for Installing Adabas with Batch/TSO

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

In this version of Adabas, the ADALNK routines are UES-enabled as distributed.

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

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

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

This section covers the following topics:

- [Non-reentrant ADALNK Batch Routine Operation](#)
- [ADALNKR: Reentrant Batch Link Routine](#)



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



## Non-reentrant ADALNK Batch Routine Operation

The ADALNK module in the Adabas 8 load library operates when the following conditions are met:

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

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

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

## ADALNKR: Reentrant Batch Link Routine

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

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

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

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



## Installing Adabas with Batch/TSO under Adabas 8

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

This section covers the following topics:

- [Supplied Modules](#)
- [Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules](#)
- [Zapping the Default Values for the Adabas 8 ADALNK or ADALNKR Modules](#)



**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 with batch/TSO under Adabas 8.

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

### Changing Default Values for the Adabas 8 ADALNK or ADALNKR Modules

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

➤ To change default values, complete the following steps:

- 1 Copy the sample member LNKGBLS (for non-reentrant links) or LNKRGBL (for reentrant links) members provided in the Adabas 8 ADA $vrs$  (where  $vrs$  is the number of the latest Adabas version delivered on the tape).SRCE library to any appropriate user source library where they can be modified. These modules contain LGBLSET parameters that are used to create default settings for link components. A complete description of LGBLSET parameters can be found in [Modifying Source Member Defaults \(LGBLSET Macro\) in Version 8](#), elsewhere in this guide.

- 2 Modify the LNKGBLS or LNKRGBL member in the user source library. Provide values for the LOGID, SVC, and other keywords to suit your installation requirements.



**Note:** The OPSYS parameter must be set to ZOS.

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

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

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

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



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

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

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

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

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

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

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

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

## 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
- LUEXIT1A: Length of LUEXIT1
- LUEXIT2A: Length of LUEXIT2
- 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
- REVIEW: Adabas Review Support
- RMI: Resource Manager Interface
- RTXNAME: Command Retry Exit Name
- SAF: Adabas Security Interface Flag
- SAP: SAP Application Support
- SAPSTR: SAP ID String
- SVCNO: Adabas SVC number
- TPMON: Operating Environment
- TRUENM: CICS TRUE Name
- UBPLOC: User Block Pool Allocation
- 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={ <u>ADACICS</u>   <i>qname</i> }

### COR: SYSCOR Exit Support

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

**DBSVCTN: DBID/SVC Routing Table**

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

**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.	DYNDBSVC={ YES   <u>NO</u> }

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

Parameter	Description	Syntax
ENTPT	<p>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.</p> <p>See also notes 1 and 2 in the installation procedure.</p>	ENTPT={ <u>ADACICS</u>   <i>name</i> }

**GBLNAME: Name of Link Globals Module**

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

**GEN: Generate CSECT or DSECT**

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

**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>   <u>YES</u> }

**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> }

**LUEXIT1A: Length of LUEXIT1**

Parameter	Description	Syntax
LUEXIT1A	The length of the work area for link user exit 1. Valid values are numbers from zero (0) through 32,767. The default is "0".	LUEXIT1A={ <i>nnn</i>   <u>0</u> }

**LUEXIT2A: Length of LUEXIT2**

Parameter	Description	Syntax
LUEXIT2A	The length of the work area for link user exit 2. Valid values are numbers from zero (0) through 32,767. The default is "0".	LUEXIT2A={ <i>nnn</i>   <i>Q</i> }

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

Parameter	Description	Syntax
LUINFO	The length of the user data to be passed to target user exit 4. Valid values are numbers from zero (0) through 32,767.  If LUINFO is not specified, the default is zero (no user save area is passed).	LUINFO={ <i>Q</i>   <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".  If LUSAVE is not specified, the default is zero (no user data is passed).	LUSAVE={ <i>72</i>   <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 CIC plus the CICS task number.</p> <p>If you run with the CICS multiple region option (MRO), you must use the default value for this option. If NETOPT=YES and MRO=YES are specified, an assembler MNOTE and a return code of 16 are produced from the assembly step.</p>	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 CICSplex environment with Natural Version 2.2.8 and above. The value is associated with all users who call the Adabas command-level link routine assembled with the specified value.</p> <p>There is no default value. If no value is specified, the Adabas internal user ID is built in the conventional manner.</p> <p>Any four-byte alphanumeric value may be specified, but it must be unique for each Adabas command-level link routine running in a CICSplex, or z/OS image. If more than one NTGPID is required (for example, both test and production Natural 2.2.8), more than one Adabas command-level link routine with associated TRUE must be generated.</p>	NTGPID= <i>4-byte-value</i>

Parameter	Description	Syntax
	If you run with the CICS multiple region option (MRO), you may use NTGPID to provide a 4-byte literal for the Adabas communication ID to be used by the Adabas SVC when multiple application regions call Adabas.	

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

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

### OPSYS: Operating System

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

### PARMTYP: Area for Adabas Parameter List

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

Parameter	Description	Syntax
	therefore recommend that CICS programs using the Adabas 8 CICS link routines use the COMMAREA only for passing data.	

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

**RENT: Reentrant Module Flag**

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

**RETRYX: Retry Command Exit Flag**

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

**REVIEW: Adabas Review Support**

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

**RMI: Resource Manager Interface**

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

**RTXNAME: Command Retry Exit Name**

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

**SAF: Adabas Security Interface Flag**

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

**SAP: SAP Application Support**

Parameter	Description	Syntax
SAP	<p>Indicates whether or not SAP user ID generation is supported.</p> <p>If SAP=YES is specified, the program will detect a SAP initialization call and set the user ID for SAP applications from the constant provided on the initialization call, plus the field ACBADD2.</p> <p>For more information, refer to the supplementary information provided to customers using the SAP application system.</p>	SAP={ <u>NO</u>   YES }

**SAPSTR: SAP ID String**

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

**SVCNO: Adabas SVC number**

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

**TPMON: Operating Environment**

Parameter	Description	Syntax
TPMON	<p>The TP monitor operating environment. Valid values should be specified as follows:</p> <ul style="list-style-type: none"> <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.</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 }

**UES: Universal Encoding Support**

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

**USERX1: User Exit 1 Flag**

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

**USERX2: User Exit 2 Flag**

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

**XWAIT: XWAIT Setting for CICS**

Parameter	Description	Syntax
XWAIT	<p>Indicates whether a standard EXEC CICS WAITCICS (XWAIT=NO) or a WAIT EVENTS EXTERNAL (XWAIT=YES) will be executed by the Adabas 8 task-related user exit (TRUE). XWAIT=YES is the default.</p> <p>The CICS WAIT EVENTS EXTERNAL (XWAIT=YES) is the recommended interface for CICS/TS 1.1 and above.</p> <p>The CICS WAITCICS statement (XWAIT=NO) is provided for use with CICS/MVS 2.1.2 and for CICS/VSE 2.1 through 2.3. It may also be used for</p>	XWAIT={ NO   <u>YES</u> }

Parameter	Description	Syntax
	<p>CICS/TS 1.1 and above, but may result in poor CICS transaction performance or unpredictable transaction results in busy CICS environments.</p> <p><b>Note:</b> If <code>XWAIT=NO</code> is specified for use under CICS/ESA 3.3, IBM APAR PN39579 must be applied to the CICS/ESA 3.3 system. For CICS/TS 1.1 and above, this APAR is not required.</p>	



#### Notes:

1. If `XWAIT=NO` is specified, the ADACICT (Adabas 8 TRUE) module issues an EXEC CICS WAIT-CICS command instead of the EXEC CICS WAIT EVENT command. `XWAIT=YES` conforms with recommended IBM usage of the WAIT and ECB lists in a high-transaction volume CICS system with CICS/TS Version 1.1 and above.
2. All EXEC CICS commands are processed by the CICS preprocessor; the LGBLSET parameters cause the subsequent assembly step to skip some of the statements.

### XWAIT Posting Mechanisms

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

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

For more details on the differences between the various CICS WAIT commands and their relationship to hard and soft posting mechanisms, consult the IBM *CICS Application Programming Reference Guide* 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.

