

z/OS



DCE Application Support Configuration and Administration Guide

z/OS



DCE Application Support Configuration and Administration Guide

Note

Before using this information and the product it supports, be sure to read the general information under Appendix I, "Notices" on page 277.

First Edition (March 2001)

This edition, SC24-5903-00, applies to Version 1 Release 1 of z/OS DCE Application Support (program number 5694-A01), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for reader's comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Information Development, Dept. G60
1701 North Street
Endicott, NY 13760-5553
United States of America

FAX (United States & Canada): 1+607+752-2327
FAX (Other Countries):
Your International Access Code +1+607+752-2327

IBMLink™ (United States customers only): GDLVME(PUBRCF)
Internet e-mail: pubrcf@vnet.ibm.com
World Wide Web: <http://www.ibm.com/servers/eserver/zseries/zos/>

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1994, 2001. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

About This Book	xiii
Who Should Use This Book	xiii
Product Names Used in This Book	xiii
How to Use This Book	xiii
Conventions Used in This Book	xiv
Where to Find More Information	xiv
Softcopy Publications	xvi
Internet Sources	xvi
Using LookAt to Look up Message Explanations	xvii
Accessing Licensed Books on the Web	xvii

Part 1. Introduction to Configuration and Administration 1

Chapter 1. Concepts and Terminology	3
What Is the Application Support Server?	3
How the Application Support Server Uses the DCE Services	7
How the Application Support Server Uses Encina Services	7
Operations and Interfaces in the Application Support Server	8
Installing and Uninstalling Interfaces	8
Attachment Facility	8
Before You Begin Configuring Your Application Support Server	9
Chapter 2. Migration Information	11
Migrating from OS/390 Release 5 or Release 6	11
Migrating from OS/390 Release 4	11
Migrating from OS/390 Release 3	11
Migrating from OS/390 Release 2	11
Migrating from IMS/ESA Release 4.1 to Release 5.1 or Above	11
Chapter 3. Overview of Administration Tasks	13
Configuring After the Initial Install: A Configuration Checklist	13
Restarting the Application Support Server	16
Changing the Application Support Server	16
Maintaining the Application Support Server	17
Installation of Application Support	18

Part 2. Configuring the Application Support Server 19

Chapter 4. Creating the Environment Variable File	21
Step 1: Create the Application Support Server Directory	21
Step 2: Specify the Directory	22
Step 3: Grant the Application Support Server Access to the Files in the Directory	23
Step 4: Create the Environment Variable File	23
Step 5: Set the Code Page of the Application Support Server	32
Chapter 5. Configuring the AS CICS Server	33
Requirements for Nontransactional and Transactional RPCs	33
CICS Resource Considerations	33
Steps for Configuring DCE AS CICS	34

Step 6: Configure Encina for Use with Application Support	37
Security Considerations in DCE AS CICS	37
Transactional Considerations	38
Chapter 6. Configuring the AS IMS OTMA Server	39
IMS Resource Considerations	39
Definitions for DCE AS IMS with OTMA	39
Step 1: Modify the IMS System Definition	40
Step 2: Ensure Environment Variables Are Set	40
Step 3: Configure Encina for Use with Application Support	41
Step 4: Set IMS OTMA Security	41
Operational Considerations When Using AS IMS with OTMA	41
Chapter 7. Configuring the AS IMS APPC Server	43
Definitions for DCE AS IMS with APPC	43
Chapter 8. Configuring the AS IMS ISC Server	47
Definitions for AS IMS with ISC	47
Chapter 9. Creating the DCE Account for the AS Server	51
Step 1: Update the Keytab File for the Application Support Server	51
Step 2: Give the AS Server Access to the DCE Endpoint Map	52
Step 3: Select the Group for the Application Support Server	52
Step 4: Create a DCE Account for the Application Support Server	52
Step 5: Ensure Certificate Lifetimes Across Cells	54
Chapter 10. Creating Administrator and User Accounts	55
Step 1: Register Users with the MVS Security Subsystem	55
Step 2: Create DCE Accounts for Application Support Server Users and Administrators	55
Step 3: Mapping DCE Principal Names to MVS User IDs	56
Chapter 11. Configuring the CDS Namespace Entries	63
Creating CDS Directories for the Application Support Server	63
Access to the Application Support Server Namespace Entries	64
Chapter 12. Giving Access to the Administration Program	67
Access to the Administration Program	67
ACL Permissions Supported for the Administration Program	68
Initial Access Control List	68
Creating the Initial ACL Database Files	68
Setting the _EUV_RPC_ACL_FILE Environment Variable	71
Switching between Current and Backup Copies of the ACL File	71
Using the ACL Editor to Give Access to the Administration Program	72
Chapter 13. Defining the Application Support Server to CDS	73
What Does Defining to CDS Mean?	73
ASUADMIN Administration Program	74
Steps in Defining the Application Support Server to CDS	74
Entering Application Support Server Attributes	76
Chapter 14. Example Configuration Worksheets	77
AS CICS Worksheet	77
AS IMS with APPC Worksheet	79
AS IMS with OTMA Worksheet	80

AS IMS with ISC Worksheet	81
Chapter 15. Blank Configuration Worksheets	83
AS CICS Worksheet	83
AS IMS with APPC Worksheet	85
AS IMS with OTMA Worksheet	86
AS IMS with ISC Worksheet	87
Chapter 16. Starting the Application Support Server	89
Startup Procedure for DCE AS CICS	89
Startup Procedure for DCE AS IMS with OTMA	90
Startup Procedure for DCE AS IMS with APPC	91
Startup Procedure for DCE AS IMS with ISC	92
<hr/>	
Part 3. Administering the Application Support Server	95
Chapter 17. Using the ASUADMIN Administration Program	97
Administration Panel Functions	98
Configuring the ASUADMIN Administration Program	99
Starting the Administration Program	99
Primary Options Panel	99
Attaching the Application Support Server	102
Stopping an Application Support Server Attachment	103
Displaying Interfaces and AS Server Operational Data	103
Installing and Uninstalling Specific Interfaces	103
Installed Interfaces That Have Been Deleted	106
Installing and Uninstalling Interfaces in Batch	107
Working with Application Support Server Definitions	108
Logging In to z/OS DCE	112
Reloading the Identity Mapping File	113
Chapter 18. Administering the AS Server Using MVS System Commands	115
Starting Attachment of an Application Support Server	115
Stopping Attachment of an Application Support Server	116
Reloading the Identity Mapping File with the MODIFY Command	116
Displaying the Server's Operational Status	117
Displaying the Status of Interfaces	119
Installing and Uninstalling Interfaces	120
Stopping the Application Support Server	120
Chapter 19. Administering the AS Server Using a Client Program	123
What You Need to Build the Client	123
vensrv_mgmt_display_if	124
vensrv_mgmt_display_ifs	126
vensrv_mgmt_display_server	128
vensrv_mgmt_if_mgmt	130
vensrv_mgmt_reload_imt	132
vensrv_mgmt_start_attachment	133
vensrv_mgmt_stop_attachment	134
Chapter 20. Using the tkadmin Command with the AS Server	135
Chapter 21. Managing Transactions Called with Transactional RPCs	137

Background on Encina Transaction Processing	139
Viewing AS Server Encina Transaction Information	144
Administering Transactions	146
Chapter 22. Problem Determination	155
Setting _EUV_SVC_MSG_LEVEL	155
Setting _EUV_SVC_MSG_LOGGING	156
Error Correlator ID	156
Using Message ASUV430I for Problem Solving in AS IMS	156
Using the ASUVPLOG Data Set to Capture IMS Messages	156
Appendix A. ASULUEE0 Exit Routine	159
Appendix B. Application Support Server Entry in CDS	167
Appendix C. Server States	169
Initializing State	169
Quiescent State	169
Starting State	170
Operational State	170
Stopping State	170
Appendix D. Interface States	173
Installed State	173
Draining State	173
Uninstalled State	173
Appendix E. ASUVSMDF IDL File	175
Appendix F. Administrative APIs Example Client Program	181
Appendix G. Application Support Server Messages and Codes	193
Understanding Application Support Server Messages	193
Severity Levels	194
Understanding DCE Status Codes	195
Understanding Message Variables	195
Messages	196
Appendix H. Codes	273
Abend Codes From DCE AS CICS	273
Server State Reason Codes	273
Appendix I. Notices	277
Trademarks	278
Programming Interface Information	278
Glossary	279
Bibliography	293
z/OS DCE Publications	293
z/OS SecureWay® Security Server Publications	293
Tool Control Language Publication	294
IBM C/C++ Language Publication	294
z/OS DCE Application Support Publications	294

Encina Publications	295
Other Publications	295
Index	297

Figures

1.	The Application Support server — A Nontransactional View	3
2.	The Application Support server — A Transactional View	6
3.	Example Startup PROC	23
4.	Example envar File for DCE AS CICS	24
5.	Example envar File for DCE AS IMS with OTMA	24
6.	Example envar File for DCE AS IMS with APPC	24
7.	Example envar File for DCE AS IMS with ISC	25
8.	LUADD statement	44
9.	Example SIADD statement	44
10.	Example APPL definition statement	45
11.	Example of an IMS TYPE and TERMINAL Definition	47
12.	Example of Defining Logical Devices	48
13.	Example of APPL Definitions in VTAM	49
14.	Example of an Entry in a VTAM Mode Table	49
15.	Example Identity Mapping File	58
16.	Example ACL File	70
17.	Create a CICS Application Support Server Definition Panel	75
18.	Create an IMS ISC Application Support Server Definition Panel	75
19.	Example Procedure to Start DCE AS CICS	90
20.	Example Procedure to Start DCE AS IMS with OTMA	91
21.	Example Procedure to Start DCE AS IMS with APPC	92
22.	Example Procedure to Start DCE AS IMS with ISC	93
23.	Primary Options Panel	100
24.	Messages on Panels	101
25.	Sample CDS Namespace Hierarchy	101
26.	Install / Uninstall Specific Interfaces Panel	104
27.	Example JCL that Runs the ASUVIFM Utility	107
28.	Work with Application Support Server Definitions Panel	108
29.	Create a CICS Application Support Server Definition Panel	109
30.	Create an IMS Application Support Server Definition Panel	110
31.	Perform DCE Login Panel	112
32.	DISPSRVR Output Format	117
33.	DISPSRVR Sample Output for AS CICS Server	118
34.	DISPSRVR Sample Output for AS IMS ISC Server	118
35.	DISPSRVR Sample Output for AS IMS APPC Server	119
36.	DISPSRVR Sample Output for AS IMS OTMA Server	119
37.	Format of Output for Displaying the Status of Interfaces	119
38.	DISPIF Sample Output for Multiple Interfaces	120
39.	Components Resolving an Encina Transaction	137
40.	The Two-Phase Commit Protocol	140
41.	Identifiers for Encina Transactions Using AS IMS	143
42.	AS IMS Failure Scenario Example for Encina Transaction	149
43.	ASULUEE0 Exit Routine	159
44.	ASUVSMDF IDL File	175
45.	Administrative APIs Example Client Program	181

Tables

1.	Criteria for Determining Which IMS Adaptor to Use	9
2.	Tasks in Restarting the Application Support Server	16
3.	Tasks in Changing the Application Support Server	17
4.	Tasks in Maintaining the Application Support Server	18
5.	Relevant Environment Variables	25
6.	Environment Variables for Application Support	26
7.	CONNECTION Attributes and Values	34
8.	SESSIONS Attributes and Values	35
9.	AS CICS Configuration Worksheet	77
10.	AS IMS with APPC Configuration Worksheet	79
11.	AS IMS with OTMA Configuration Worksheet	80
12.	AS IMS with ISC Configuration Worksheet	81
13.	AS CICS Configuration Worksheet	83
14.	AS IMS with APPC Configuration Worksheet	85
15.	AS IMS with OTMA Configuration Worksheet	86
16.	AS IMS with ISC Configuration Worksheet	87
17.	Administrative Tasks and Ways to Do Them	97
18.	Action Code 1: Responses and Meanings	105
19.	Action Code 0: Responses and Meanings	106
20.	Phases and States of a Transaction	141
21.	Action to Perform Based on Transaction State	151
22.	CDS Attributes of the Application Support Server	167

About This Book

Configuring and administering the Application Support (AS) server on the z/OS host system involves tasks that are performed on the Customer Information Control System (CICS®) or Information Management System (IMS™) subsystem, as well as on the z/OS Distributed Computing Environment (DCE). The configuration tasks are performed after installing the Application Support server to make it operational. Administration tasks are performed regularly to maintain the Application Support server. This book describes how to configure and administer the Application Support server and the facilities that are used to accomplish these tasks.

Who Should Use This Book

This book is intended for system and network administrators who understand the basic concepts of the Distributed Computing Environment (DCE) and know how to use the administration facilities that z/OS DCE provides.

Knowledge of z/OS DCE and CICS or IMS is also assumed in the discussions in this book.

Administrators should be familiar with DCE concepts and use of DCE in their environments. The following books provide background information about DCE:

- *z/OS DCE Introduction*, GC24-5911
- *z/OS DCE Administration Guide*, SC24-5904

Product Names Used in This Book

References in this book to “MVS” refer to the MVS element of z/OS.

Any reference to DCE in this book is understood specifically to mean z/OS DCE unless otherwise noted.

The name **DCE AS CICS** in this book refers to the server that accesses the CICS region. The name **DCE AS IMS** in this book refers to the server that accesses the IMS region. The name **Application Support server** in this book is a generic name for either the AS CICS or AS IMS server.

In this book the term “DCE Security Server” (or simply “Security Server”) refers to the z/OS SecureWay® Security Server DCE or to a DCE Security Server provided on another host in the DCE cell. The z/OS SecureWay Security Server DCE is a component of the SecureWay Security Server for z/OS.

How to Use This Book

This book is divided into three parts. The first part introduces the concepts of the Application Support server and provides an overview of the configuration and administration tasks that you perform.

The second part describes in detail the tasks that you perform to configure and start the Application Support server.

The third part describes the tasks that are involved in the administration of the Application Support server.

This book also contains the Application Support server codes and messages that can be used for handling error situations. The list of codes and messages, including explanations, are in the appendix of this book.

Conventions Used in This Book

This book uses the following typographic conventions:

Bold	Bold words or characters represent system elements that you must enter into the system literally, such as commands, options, or path names.
<i>Italic</i>	<i>Italic</i> words or characters represent values for variables.
Example font	Examples and information displayed by the system appear in constant width type style.
[]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.
	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
...	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.
\	A backslash is used as a continuation character when entering commands from the shell that exceed one line (255 characters). If the command exceeds one line, use the backslash character \ as the last non-blank character on the line to be continued, and continue the command on the next line.

This book uses the following keying conventions:

<Alt-c>	The notation <Alt-c> followed by the name of a key indicates a control character sequence.
<Return>	The notation <Return> refers to the key on your keyboard that is labeled with the word Return or Enter, or with a left arrow.
Entering commands	When instructed to enter a command, type the command name and then press <Return>.

Where to Find More Information

Where necessary, this book references information in other books using shortened versions of the book title. For complete titles and order numbers of the books for all products that are part of z/OS, see the *z/OS Information Roadmap*, SA22-7500. For complete titles and order numbers of the books for z/OS DCE, refer to the publications listed in the “Bibliography” on page 293.

For information about installing z/OS DCE Application Support components, see the *z/OS Program Directory*.

For information about using z/OS UNIX System Services commands to perform end-user tasks, see *z/OS UNIX System Services User's Guide*, SA22-7801.

For information about developing client applications for the Application Support server, see *z/OS DCE Application Support Programming Guide*.

For information about planning and configuring z/OS DCE components, see:

- *z/OS DCE Planning*

- *z/OS DCE Configuring and Getting Started*

For information about z/OS DCE administrative tasks and facilities and administrative commands and syntax, see:

- *z/OS DCE Administration Guide*
- *z/OS DCE Command Reference*

For error message information, see:

- *z/OS DCE Application Development Reference*
- *z/OS DCE Messages and Codes*

For information about DCE administration on other IBM systems, see the administrator's guide for those systems.

The following lists provide titles and order numbers for books related to other products.

For information about CICS, see:

- *CICS Resource Definition Guide*, SC34-5722
- *CICS External Interfaces Guide*, SC34-5709
- *CICS Supplied Transactions*, SC34-5724
- *CICS System Programming Reference*, SC34-5726
- *CICS Recovery and Restart Guide*, SC34-5721
- *CICS Transaction Server for OS/390 Installation Guide*, GC34-5697

For information about using IMS/ESA,[®] see:

- *IMS/ESA Administration Guide: System*, SC26-8730
- *IMS/ESA Administration Guide: Transaction Manager*, SC26-8731
- *IMS/ESA Operator's Reference*, SC26-8742
- *IMS/ESA Open Transaction Manager Access Guide*, SC26-8743
- *IMS/ESA Customization Guide*, SC26-8732
- *IMS/ESA Operations Guide*, SC26-8741

For information about using a sysplex, see:

- *z/OS MVS Programming: Sysplex Services Guide*, SA22-7617
- *z/OS MVS Programming: Sysplex Services Reference*, SA22-7618
- *z/OS MVS Setting Up a Sysplex*, SA22-7625
- *z/OS MVS Programming: Resource Recovery*, SA22-7616

For information about RACF,[®] a component of the SecureWay Security Server for z/OS, see:

- *z/OS SecureWay Security Server RACF Security Administrator's Guide*, SA22-7683
- *z/OS SecureWay Security Server RACF Command Language Reference*, SA22-7687
- *CICS RACF Security Guide*, SC34-5720

(The previously mentioned *z/OS DCE Administration Guide* describes RACF interoperability and *CICS External Interfaces Guide* provides additional information about RACF security.)

For information about VTAM,® see:

- *OS/390 IBM Communications Server: SNA Programming*, SC31-8573
- *z/OS Communications Server: SNA Network Implementation Guide*, SC31-8777
- *z/OS Communications Server: SNA Resource Definition Reference*, SC31-8778
- *IBM Planning for NetView,® NCP, and VTAM*, SC31-8063

For information about APPC management and security, see:

- *z/OS MVS Planning: APPC/MVS Management*, SA22-7599

For information about requirements for the MVS System Logger Component, see the *z/OS MVS Programming: Assembler Services Guide*, SA22-7605.

For information about how to configure DCE for use with the Encina Toolkit and administer an Encina Toolkit server, see the Encina Toolkit Administrator's Guide for your system.

For more information about administering the Encina Toolkit on z/OS, see:

- *z/OS Encina Toolkit Executive Guide and Reference*, SC24-5919
- *z/OS Encina Transactional RPC Support for IMS*, SC24-5920

For information about environment variables for setting code pages, see *z/OS C/C++ Programming Guide*, SC09-4765.

For help in handling errors, see:

- *CICS Data Areas*, LY33-6096
- *z/OS Language Environment Debugging Guide*, GA22-7560
- *z/OS ISPF User's Guide*, SC34-4822
- *z/OS ISPF Services Guide*, SC34-4819
- *z/OS ISPF Dialog Developer's Guide and Reference*, SC34-4821

Softcopy Publications

The z/OS DCE Application Support library is available on a CD-ROM, *z/OS Collection*, SK3T-4269. The CD-ROM online library collection is a set of unlicensed books for z/OS and related products that includes the IBM Library Reader.™ This is a program that enables you to view the BookManager® files. This CD-ROM also contains the Portable Document Format (PDF) files. You can view or print these files with the Adobe Acrobat reader.

Internet Sources

The Softcopy z/OS publications are also available for web-browsing and for viewing or printing PDFs using the following URL:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/>

You can also provide comments about this book and any other z/OS documentation by visiting that URL. Your feedback is important in helping to provide the most accurate and high-quality information.

Using LookAt to Look up Message Explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages. You can also use LookAt to look up explanations of system abends.

Using LookAt to find information is faster than a conventional search because LookAt goes directly to the explanation.

LookAt can be accessed from the Internet or from a TSO command line.

You can use LookAt on the Internet at:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>

To use LookAt as a TSO command, LookAt must be installed on your host system. You can obtain the LookAt code for TSO from the LookAt Web site by clicking on the **News and Help** link or from the *z/OS Collection*, SK3T-4269.

To find a message explanation from a TSO command line, simply enter: **lookat** *message-id* as in the following:

```
lookat iec192i
```

This results in direct access to the message explanation for message IEC192I.

To find a message explanation from the LookAt Web site, simply enter the message ID and select the release with which you are working.

Note: Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS MVS Routing and Descriptor Codes*, SA22-7624. For such messages, LookAt prompts you to choose which book to open.

Accessing Licensed Books on the Web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link site:

<http://www.ibm.com/servers/resourceLink>

Licensed books are available only to customers with a z/OS license. Access to these books requires an IBM Resource Link user ID, password, and z/OS licensed book key code. The z/OS order that you received provides a memo that includes your key code.

To obtain your IBM Resource Link user ID and password, logon to:

<http://www.ibm.com/servers/resourceLink>

To register for access to the z/OS licensed books:

1. Logon to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.
3. Select **Access Profile**.
4. Select **Request Access to Licensed books**.
5. Supply your key code where requested and select the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed.

After your request is processed you will receive an e-mail confirmation.

Note: You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

To access the licensed books:

1. Logon to Resource Link using your Resource Link user ID and password.
2. Select **Library**.
3. Select **zSeries**.
4. Select **Software**.
5. Select **z/OS**.
6. Access the licensed book by selecting the appropriate element.

Part 1. Introduction to Configuration and Administration

This part of the book introduces the basic concepts of the Application Support server. It also provides an overview of typical configuration and administration tasks that are performed on the Application Support server.

Chapter 1. Concepts and Terminology	3	Chapter 2. Migration Information	11
What Is the Application Support Server?	3	Migrating from OS/390 Release 5 or Release	
For Nontransactional RPCs	3	6	11
For Transactional RPCs	4	Migrating from OS/390 Release 4	11
How the Application Support Server Uses the		Migrating from OS/390 Release 3	11
DCE Services	7	Migrating from OS/390 Release 2	11
The Application Support Server and RPC	7	Migrating from IMS/ESA Release 4.1 to	
The Application Support Server and DCE		Release 5.1 or Above	11
Security Service	7		
The Application Support Server and DCE		Chapter 3. Overview of Administration	
Directory Services	7	Tasks	13
How the Application Support Server Uses		Configuring After the Initial Install: A	
Encina Services	7	Configuration Checklist	13
Encina Support for IMS	8	Using Sample JCL to Automate	
Operations and Interfaces in the Application		Configuration Steps	15
Support Server	8	Restarting the Application Support Server	16
Installing and Uninstalling Interfaces	8	Changing the Application Support Server	16
Attachment Facility	8	Maintaining the Application Support Server	17
Before You Begin Configuring Your Application		Installation of Application Support	18
Support Server	9		

Chapter 1. Concepts and Terminology

This section introduces you to the concepts of the Application Support server and to the terminology used in this book, and provides you with information you should be aware of before you begin configuring the Application Support server.

What Is the Application Support Server?

The Application Support server is a z/OS DCE server that enables client applications anywhere in DCE to use the resources of CICS and IMS.

The Application Support server facilitates the interaction between DCE clients and the CICS or IMS regions. It transforms the requests from DCE clients into a form that the CICS or IMS regions can understand. It also makes the CICS or IMS regions appear like regular RPC-based servers to DCE clients. The AS server can handle DCE RPCs and transactional RPCs.

Clients are either applications residing on non-MVS hosts (for example, UNIX workstations), or z/OS applications that do not reside in CICS or IMS regions. For a transactional RPC, clients use Encina semantics, making the call within an Encina transaction.

For Nontransactional RPCs

The client application performs a remote procedure call (RPC), using DCE's RPC facility, to CICS or IMS transaction programs through the Application Support server. The CICS or IMS programs perform the requested operation and the result is returned to the client through the Application Support server. Figure 1 shows the relationship between DCE clients and the Application Support server and communication paths between the DCE client, the Application Support server, and the CICS or IMS regions.

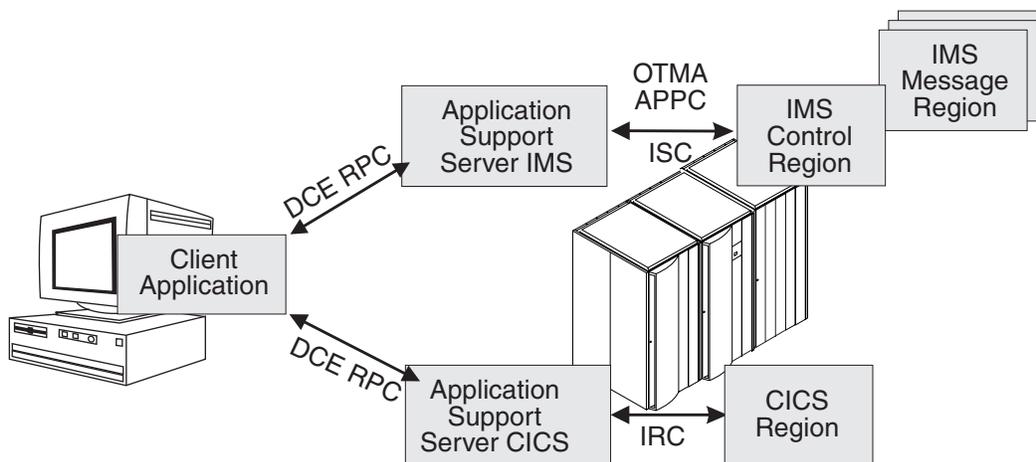


Figure 1. The Application Support server — A Nontransactional View

Each Application Support server can attach to one and only one CICS or IMS control region. Multiple Application Support servers can attach to a single CICS or IMS subsystem.

For Transactional RPCs

AS transactional RPC support for IMS and CICS transactions lets AS IMS OTMA servers and AS CICS servers participate in Encina¹ distributed transaction processing applications. (See “Before You Begin Configuring Your Application Support Server” on page 9 for information about requirements for configuring your Application Support server.) An Encina *transaction* is a set of operations that must be done as a single unit for consistent transformation of data. In a typical Encina transaction processing scenario, a client application begins a transaction, accesses some number of remote servers that manage recoverable data, and then ends the transaction.

AS transactional support lets the client application include IMS and CICS transactions within the scope of an Encina transaction. Data integrity is guaranteed across all servers (or participants) involved. If any participant, including AS IMS or AS CICS, is unable to perform the required operations, all of the work is undone (aborted or rolled back). Updates to recoverable data are committed only if all participants can successfully perform their work. Once committed, updates are not lost in subsequent system failures.

OTMA is a transaction-based, connectionless client-server protocol for IMS/ESA in an MVS sysplex environment. You can use OTMA for nontransactional RPCs. For AS IMS, it is required for transactional RPCs. See Chapter 6, “Configuring the AS IMS OTMA Server” on page 39 for more information about OTMA.

The z/OS Encina Toolkit Executive enhances DCE with transactional semantics. It implements a two-phase commit protocol. This is a set of actions that ensures an application program makes *all* the changes of a transaction to a collection of resources or makes *no* changes. The z/OS Encina Toolkit Executive is a subset of Transarc's Encina.

Note: In this section, the term *transaction* refers to an Encina transaction (unless specifically indicated to be an IMS or CICS transaction).

Transactions exhibit the following properties (called the ACID properties):

- **Atomicity.** A transaction is either successful or unsuccessful. Either all of the operations that make up a transaction take effect or none take effect. A successful transaction is said to commit. An unsuccessful transaction is said to abort. Any operations performed by an aborted transaction are undone.
- **Consistency.** A transaction transforms distributed data from one consistent (correct) state to another.
- **Isolation.** The transactions appear to be sequential, acting as though one completed before the other began, even though they may run concurrently. The effects of a given transaction are not visible to other transactions until that transaction commits.
- **Durability (also known as permanence).** Once a transaction commits, its effects are permanent. A subsequent failure (such as abnormal program termination, communications failure, or hardware crash) does not alter the transaction's effects.

You can use Encina services and transactional RPCs to group operations into transactions. This ensures that each transaction is performed atomically and independently of other transactions, and that, once completed, its effects are permanent. Successfully completing a transaction is called *commitment*.

The z/OS Encina Toolkit Executive enhances DCE RPCs with transactional semantics by implementing a two-phase commit protocol that synchronizes related pieces of work taking place in different processes. The protocol guarantees that all the processes successfully complete the work or that it is not performed

¹ Encina is a family of products that provides a range of services for developing and deploying large scale client-server and open, distributed transaction processing systems. Layered on top of OSF DCE, Encina uniquely extends DCE with services such as transactions that are necessary when building business-critical systems.

at all. For example, changes to data either all fail or all succeed. The goal is to ensure that each participant in a transaction, that is, each resource manager updating data in the transaction, takes the same action (commits or aborts).

Until a participant is prepared, it can unilaterally decide to abort the transaction. Once all of the work of the transaction is complete, the application attempts to commit the work by invoking the two-phase commit protocol. In the first phase, the prepare phase, each participant is asked to prepare. Once a participant is prepared, it agrees to accept whatever outcome the coordinator decides on (it can no longer unilaterally abort the transaction).

The final outcome of a transaction (how the transaction ends) depends upon the individual outcome of each participant in the transaction. In the resolution phase, a transaction commits if all participants can commit or aborts if any one participant aborts.

The prepare phase consists of all actions up until the time each participant notifies the coordinator that it is ready to commit (or to abort). The resolution phase begins when the coordinator instructs each participant to either commit or abort and ends when all participants have completed their work.

Clients are Encina applications. AS supports transactional RPCs only for AS CICS servers and for AS IMS OTMA servers. (There is no transactional support for AS IMS APPC or AS IMS ISC servers.) The Encina application performs a transactional RPC to a transaction program through the Application Support server, and the program performs the requested operation and returns the result to the client through the Application Support server. For transactional RPCs, the Application Support server is also connected with the Resource Recovery Services (RRS) component of Recoverable Resource Management Services (RRMS). This provides locking, logging, recovery, and other services. Figure 2 on page 6 shows the relationship between DCE clients and the Application Support server and communication paths between the Encina client, the Application Support server, and IMS or CICS.

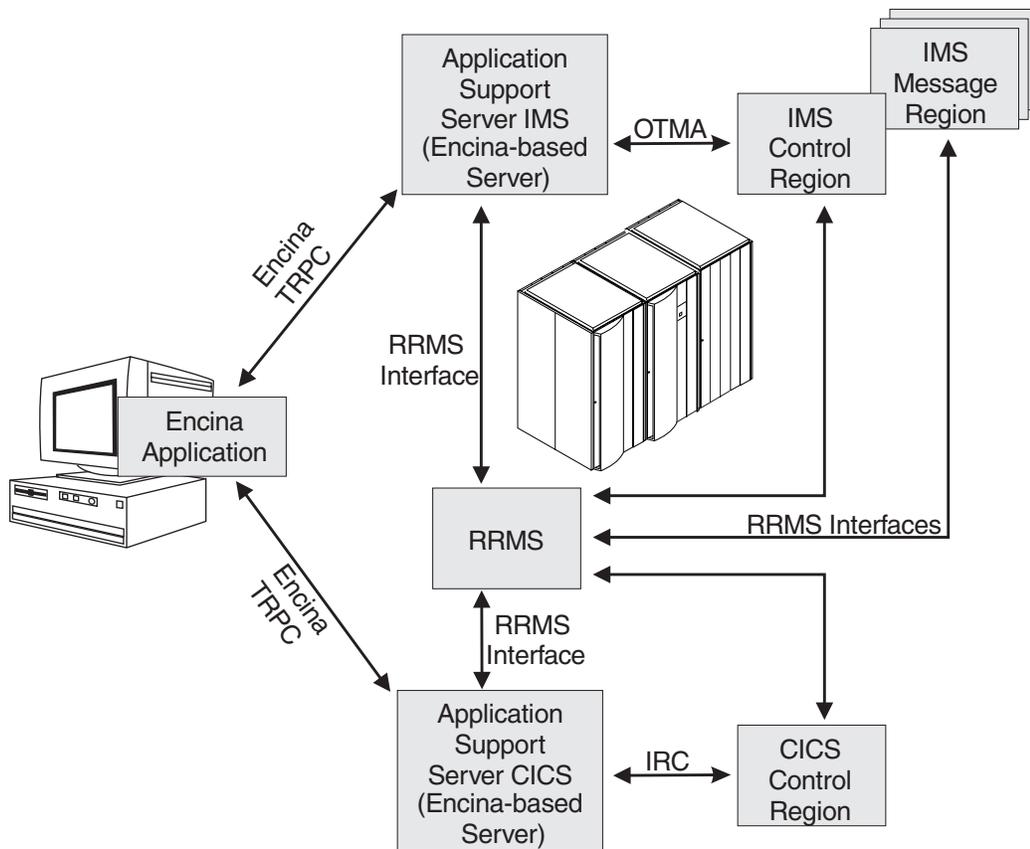


Figure 2. The Application Support server — A Transactional View

Each Application Support server can attach to only one IMS or CICS control region. Multiple Application Support servers can attach to a single IMS or CICS control region. In a sysplex, AS IMS and the IMS subsystem—or AS CICS and the CICS subsystem—must be on the same CPU.

How the Application Support Server Uses the DCE Services

This section describes the relationship between the Application Support server and the essential DCE services, namely, RPC, Security Service, and Directory Service.

The Application Support Server and RPC

Using the client-server paradigm, the CICS or IMS transaction programs may be thought of as the ultimate **servers**, being the programs that provide the services that the DCE clients request. Application programs calling the CICS or IMS transaction programs are the **clients** that use RPC to make requests to the CICS or IMS transaction programs, through the Application Support server.

The Application Support Server and DCE Security Service

The Application Support server interacts with the DCE Security Service in the same manner as any DCE server. The Application Support server runs under its own identity and is subject to the authentication and authorization by the Security Service. The Application Support server must have its own DCE account as well as the appropriate privileges to all DCE resources that it needs to access.

The DCE Security Service authenticates and authorizes clients of the Application Support server. All Application Support server users must have a DCE account as well as the appropriate privileges to access the Application Support server.

The Application Support Server and DCE Directory Services

The Application Support server interacts with the Directory service in a variety of ways. In DCE, servers need to export their interfaces to the clients through DCE Directory Service. Similarly, information about the existence of CICS or IMS transaction programs must be exported to all clients through the Directory Service.

The Cell Directory Service also stores configuration information the Application Support server uses.

How the Application Support Server Uses Encina Services

The AS IMS OTMA server or the AS CICS server can be an Encina-based server if you are using transactional RPCs. As an Encina-based server, AS IMS or AS CICS and customer-built transactional stubs use functions in the z/OS Encina Toolkit Executive. The z/OS Encina Toolkit Executive is packaged as a Dynamic Link Library (DLL). If you are using transactional RPCs, you must ensure that the z/OS Encina Toolkit Executive DLL is available when you run your AS IMS OTMA or AS CICS server.

Although the z/OS Encina Toolkit Executive supports only ephemeral clients (ones with no facility for logging results or recovery), Application Support provides recovery by using native z/OS RRS and IMS or CICS facilities in combination with Toolkit services.

Application Support uses Encina semantics. Therefore, you can (within the scope of an Encina transaction) coordinate IMS or CICS updates with updates to other types of resources that Encina supports on other systems. (An example would be the Structured File Server (SFS), which provides log-based recovery.)

Encina Support for IMS

See *z/OS Encina Transactional RPC Support for IMS* for information about this support.

Operations and Interfaces in the Application Support Server

In DCE, an operation is a task that is performed by a routine or a procedure offered by a DCE server. In the Application Support server context, an operation corresponds to a CICS or IMS transaction program.

In DCE, an interface is a collection of operations that a client can call through RPC. In the Application Support server context, an interface represents a collection of callable CICS or IMS transaction programs.

Installing and Uninstalling Interfaces

A DCE server must register the interfaces that it supports with DCE. Typically, this action is performed when the server is started and initialized. The Application Support server is different from typical DCE servers: the interfaces that it supports are actually CICS or IMS transaction programs that are not part of the Application Support server code. The Application Support server does not know beforehand which interfaces it must register. Thus, it cannot export its supported interfaces when it is started and initialized. This task is known as **installing** the interfaces. Unless the interface definition must change, each interface needs to be installed only once per server. The interface descriptions are stored in the CDS associated with the server name. When the server is restarted, it reads these descriptions and prepares to service each interface.

The administrator can install interfaces in several ways:

- Through the administration panels the Application Support product provides
- In batch using the **ASUVIFM** utility
- With MVS MODIFY system command support
- By writing a client using the API `vensrv_mgmt_if_mgmt`.

Uninstalling an interface deregisters the interface from DCE.

If you are using partitioned data set extended (PDSE), you can dynamically install and uninstall interfaces without restarting the Application Support server.

Attachment Facility

The Attachment Facility is that part of the Application Support server that provides support for RPC clients to access CICS or IMS programs. The Attachment Facility initializes the communication links between the Application Support server and the CICS or IMS regions.

Before You Begin Configuring Your Application Support Server

Before you begin the task of configuring your Application Support server, you should do the following:

- Read this chapter to familiarize yourself with Application Support server basics.
- If you are migrating from an earlier release of Application Support (rather than installing it for the first time), read Chapter 2, “Migration Information” on page 11 for information you may need.
- Read Chapter 3, “Overview of Administration Tasks” on page 13. This outlines the tasks you must perform to configure and administer your Application Support server.
- Determine which communications method you will use to pass data between the Application Support server and IMS or CICS.
 - For CICS transactional RPC support, you need the CICS Transaction Server 1.3 or later.
 - For CICS nontransactional RPC support, you need CICS/ESA® 4.1 or later.
 - For IMS there are three methods of communication available between the Application Support server and IMS:
 - OTMA
 - APPC
 - ISC

You should choose one method and plan to stay with it because changing methods generally requires programming changes.

Table 1. Criteria for Determining Which IMS Adaptor to Use

Adaptor	IMS Version	Support for Transactional RPCs	Existing ISC Programs
OTMA	6.1 or later	yes or no	no
APPC	4 or later	no	no
ISC	3.1 or later	no	yes

For details about OTMA, see page 39. For details about APPC, see page 43. For details about ISC, see page 47.

- Fill in the values in the example configuration worksheets in Chapter 15, “Blank Configuration Worksheets” on page 83. Meet with your system administrator and fill in these values together before you begin configuring. Chapter 14, “Example Configuration Worksheets” on page 77 contains worksheets that are filled in with the example values used in this book.

Chapter 2. Migration Information

This chapter highlights topics important for migrating from OS/390.[®] If you are migrating from OS/390 Release 7 or later, there are no migration actions for z/OS. If you are migrating to z/OS from an earlier release of OS/390, follow the migration actions specified. The requirements listed for each release are cumulative—that is, perform migration actions starting with the release from which you are migrating.

Migrating from OS/390 Release 5 or Release 6

If you are using the Customization Component for an AS IMS OTMA or AS CICS server, you must rebuild the server module. You also need to change your JCL.

Migrating from OS/390 Release 4

No changes to your Application Support server definitions or programs are required.

Migrating from OS/390 Release 3

You can install Application Support interfaces that were created using OS/390 Release 3 or previous releases without changing them. However, if you recompile or relink your interfaces using z/OS or OS/390 Release 5 or later, you must change your link-edit step to reference **EUVPDLL.x**. See the *z/OS DCE Application Development Guide: Introduction and Style* for more information about this file.

When using the ASUADMIN administration program, you must use the same levels of the Application Support server, ASUADMIN program, ASUADMIN ISPF panels, and ASUVIFM utility. For example, you cannot run the ASUADMIN program, ASUADMIN ISPF panels, or ASUVIFM utility with an OS/390 Release 2 or earlier Application Support server and vice versa.

Migrating from OS/390 Release 2

In addition to the changes “Migrating from OS/390 Release 3” describes, if you are using the APPC adaptor, be aware that APPC after Release 2 processes LU-LOCAL differently than in earlier releases. Therefore, RACF changes may be necessary. See Chapter 7, “Configuring the AS IMS APPC Server” on page 43 for details.

Migrating from IMS/ESA Release 4.1 to Release 5.1 or Above

If you are using the APPC adaptor, the DFSLUEE0 IMS exit customization that Application Support provides to work with IMS/ESA Release 4.1 must be changed to work with IMS/ESA 5.1 or higher.

Chapter 3. Overview of Administration Tasks

The chapter gives an overview of the important configuration and administration tasks for an Application Support server. It lists the tasks for four different scenarios. Administration scenarios vary from one site to another. These lists are guides that you can change according to the specific conditions in your site.

The tables in this chapter point to the sections in this book describing specific administration tasks in detail.

Configuring After the Initial Install: A Configuration Checklist

After the installation of the Application Support server, you have to perform the following tasks to make the Application Support server operational:

Step 1. Create the environment variable (envar) file for the Application Support server.

Each instance of the Application Support server must have its own **envar** file to store its own environment variables. Each Application Support server must have its own directory.

See Chapter 4, “Creating the Environment Variable File” on page 21.

Step 2. Configure for CICS or IMS usage.

You must configure the CICS or IMS subsystem to which the Application Support server is going to be attached to ensure that proper communication links are established. You define specific resources to CICS System Definition file or IMS resource attribute tables. If you are using IMS with APPC you can customize the DFSLUEE0 exit code to enable Application Support to accept MOD Name and LTERM data. (OTMA can determine this information dynamically and, therefore, does not need the DFSLUEE0 exit.)

If you are using IMS with OTMA or CICS for transactional RPCs, you must enable the IMS or CICS subsystem to communicate with the MVS Cross System Coupling Facility (XCF). Also, ensure that RRMS and the z/OS Encina Toolkit Executive are available.

If you are using CICS, see Chapter 5, “Configuring the AS CICS Server” on page 33. If you are using OTMA, see Chapter 6, “Configuring the AS IMS OTMA Server” on page 39. If you are using APPC, see Chapter 7, “Configuring the AS IMS APPC Server” on page 43. If you are using ISC, see Chapter 8, “Configuring the AS IMS ISC Server” on page 47.

Step 3. Create the DCE account of the Application Support server.

The Application Support server must run under its own DCE identity. The procedure for creating this account involves adding information in the Security registry and creating a local file, called the **keytab** file.

Note: Someone with DCE administrator privileges must perform this step.

See Chapter 9, “Creating the DCE Account for the AS Server” on page 51.

Step 4. Create the DCE account of the Application Support server administrator and the DCE users of the Application Support server and register users with DCE and MVS.

The Application Support server administrator is a privileged DCE user who has access to the Application Support server administration program.

Note: Someone with DCE administrator privileges must perform this step.

See Chapter 10, “Creating Administrator and User Accounts” on page 55.

Step 5. Map DCE principal names to local MVS user IDs.

Creating DCE user accounts for access to the Application Support server also includes establishing a relationship between these two identities. This can be done using the Identity Mapping file or with RACF.

See “Step 3: Mapping DCE Principal Names to MVS User IDs” on page 56.

Step 6. Configure the CDS namespace entries of the Application Support server.

You must manually create the CDS directories to hold the object entries of the Application Support servers using the CDS control program. You must set up access to the namespace entries of the Application Support server to allow access for both the Application Support server administrator and the Application Support server principal.

Note: Someone with DCE administrator privileges must perform this step.

See Chapter 11, “Configuring the CDS Namespace Entries” on page 63.

Step 7. Give the Application Support server administrator access to the administration program.

The Access Control List (ACL) of the Application Support server determines access to the administration program (and to the Encina **tkadmin** command—see page 135 for details). This step involves creating an initial ACL for the Application Support server that has an entry for the Application Support server administrator and then setting the `_EUV_RPC_ACL_FILE` environment variable to point to the initial ACL file.

Note: Someone with DCE administrator privileges must perform this step.

See Chapter 12, “Giving Access to the Administration Program” on page 67.

Step 8. Configure the ASUADMIN administration program.

The ASUADMIN administration program lets you perform administrative functions through ISPF panels. Before you can use ASUADMIN, you must configure it.

See “Configuring the ASUADMIN Administration Program” on page 99.

Step 9. Define the Application Support server to the Cell Directory Service.

The Application Support server must have an entry in the Cell Directory Service namespace. The Application Support server advertises the interfaces that it supports in CDS by providing binding information to these interfaces. This entry also contains CICS-related or IMS-related attributes that the Application Support server uses when it is started and initialized.

You define the Application Support server to CDS through the Application Support server's administration program, ASUADMIN.

Note: Someone with DCE administrator privileges must perform this step.

See Chapter 13, “Defining the Application Support Server to CDS” on page 73 and “Step 3: Configure Encina for Use with Application Support” on page 41.

Step 10. If you are using an AS IMS OTMA server or an AS CICS server for transactional RPCs, start RRS. RRS must be installed, configured, and operational before starting the Application Support server. For more information, see *z/OS MVS Programming: Resource Recovery*.

Step 11. Build the example **Min-Max** application that is provided with this product to exercise the Application Support server.

The file `/usr/lpp/dceas/examples/README.FIRST` provides directions for the installation verification program (IVP) and sample programs for Application Support. Before reading any README file associated with a particular IVP variation, read this file for instructions on how to build and run the IVP.

Step 12. Start the Application Support server.

See Chapter 16, “Starting the Application Support Server” on page 89.

Step 13. Install interfaces.

You must install interfaces that represent the callable CICS or IMS transaction programs on the Application Support server. Interfaces are built as part of the application development process. A sample application (**Min-Max**) is available with the Application Support server. *z/OS DCE Application Support Programming Guide* discusses building interfaces.

When an interface is installed, it is registered to the CDS namespace and to the DCE endpoint map.

You can install interfaces using the administration panels that the Application Support server administration program provides. You can also use the **ASUVIFM** utility to install interfaces in batch mode or the **MODIFY** command. See Chapter 17, “Using the ASUADMIN Administration Program” on page 97, “Installing and Uninstalling Specific Interfaces” on page 103, or “Installing and Uninstalling Interfaces” on page 120.

Step 14. Attach the Application Support server to the CICS or IMS subsystem.

Attaching the Application Support server to the CICS or IMS subsystem establishes the communication links between these two systems. This can be done in three ways: through the administration panels, by entering a system command, or automatically each time the server is started by setting the `_ASU_AUTOSTART` environment variable.

See Chapter 17, “Using the ASUADMIN Administration Program” on page 97, “Attaching the Application Support Server” on page 102, or Chapter 18, “Administering the AS Server Using MVS System Commands” on page 115.

Using Sample JCL to Automate Configuration Steps

A sample JCL supplied with Application Support performs some of the steps outlined in “Configuring After the Initial Install: A Configuration Checklist” on page 13. The recommended procedure for configuring the Application Support servers is to follow the instructions the rest of this book describes. This will help you understand what is required to configure and start the Application Support servers. You may choose to use the JCL if:

- You are having difficulty performing the manual steps required during a normal configuration process and are unable to determine the cause.
- You are in a test or development mode and need to create new servers often.
- You find it necessary to contact IBM Software Support for a software problem. In this case, you may be asked to run the sample JCL to verify configuration.

Note: Before running the JCL, ensure that the `cds_attributes` file in the `/opt/dcelocal/etc` directory on your system has the following two lines:

```
1.3.18.0.2.4.3 Installed_IFs      char
1.3.18.0.2.4.4 SubSystem_Info   char
```

If these lines do not appear in the file, add them to it. The sample JCL is supplied as four members in the SASUSAMP data set:

```
ASPREFX.SASUSAMP(ASUCFCIC) - the JCL for configuring AS CICS
ASPREFX.SASUSAMP(ASUCFIMI) - the JCL for configuring AS IMS with ISC
ASPREFX.SASUSAMP(ASUCFIMA) - the JCL for configuring AS IMS with APPC
ASPREFX.SASUSAMP(ASUCFIMO) - the JCL for configuring AS IMS with OTMA
```

`ASPREFX` is the high-level qualifier of the MVS data set where your Application Support files are installed. The JCL matches the examples shown in this book and the information shown in the example configuration worksheets in Chapter 14, “Example Configuration Worksheets” on page 77. You **must**

change the JCL using the information for your specific installation from the configuration worksheets you fill out in Chapter 15, “Blank Configuration Worksheets” on page 83. The JCL performs only portions of the required configuration. If you choose to change and use the JCL, you still need to perform the following steps manually:

- Configure the CICS or IMS subsystem as shown in Chapter 5, “Configuring the AS CICS Server” on page 33, Chapter 6, “Configuring the AS IMS OTMA Server” on page 39, Chapter 7, “Configuring the AS IMS APPC Server” on page 43, or Chapter 8, “Configuring the AS IMS ISC Server” on page 47.
- Change the envar file as shown in Chapter 4, “Creating the Environment Variable File” on page 21.
- Create the Identity Mapping file or use RACF to map DCE principal names to MVS user IDs as described in “Step 3: Mapping DCE Principal Names to MVS User IDs” on page 56.
- Create the ACL database file as shown in “Creating the Initial ACL Database Files” on page 68.

Restarting the Application Support Server

You may need to stop and then restart the Application Support server for a variety of reasons. (In particular, if the Encina two-phase commit processing was occurring when the server stopped, it is very important that you restart the Application Support server as quickly as possible. This is because any locks held by the transactions — on all systems involved in the distributed transaction — remain until the Application Support server is restarted.) When restarting the Application Support server, you have to perform the following tasks to make the Application Support server operational again:

1. Start the Application Support server.
2. Attach the Application Support server to the CICS or IMS subsystem.

After restarting the Application Support server, you must attach the Application Support server to the CICS or IMS subsystem again. You can use the `_ASU_AUTOSTART` environment variable to cause the Application Support server to attach to the CICS or IMS subsystem automatically each time the server is started. See Chapter 4, “Creating the Environment Variable File” on page 21 for more information on environment variables.

Table 2 indicates the sections in this book where these tasks are described in more detail.

<i>Table 2. Tasks in Restarting the Application Support Server</i>	
Task	Book Section
Starting the Application Support server	Chapter 16, “Starting the Application Support Server” on page 89.
Attaching the Application Support server to CICS or IMS	Chapter 17, “Using the ASUADMIN Administration Program” on page 97, “Attaching the Application Support Server” on page 102, or Chapter 18, “Administering the AS Server Using MVS System Commands” on page 115.

Changing the Application Support Server

There may be instances when you have to change the definition of the Application Support server. A typical example is moving the Application Support server to another CICS or IMS subsystem. To change the definition of the Application Support server, perform the following:

1. Reconfigure the CICS or IMS subsystem.

When you change the definition of an Application Support server, some CICS or IMS resource attributes are changed. These changes must be reflected in the CICS or IMS resource definitions.

2. Change the definition of the Application Support server.
3. Start the Application Support server.
4. If necessary (for example, if you move the server to another machine), install new interfaces.
5. Attach the Application Support server to the CICS or IMS subsystem.

Table 3 indicates the sections in this book where these tasks are described in more detail.

<i>Table 3. Tasks in Changing the Application Support Server</i>	
Task	Book Section
Reconfiguring CICS or IMS	Chapter 5, “Configuring the AS CICS Server” on page 33, Chapter 6, “Configuring the AS IMS OTMA Server” on page 39, Chapter 7, “Configuring the AS IMS APPC Server” on page 43, or Chapter 8, “Configuring the AS IMS ISC Server” on page 47.
Changing the Application Support server definition	Chapter 13, “Defining the Application Support Server to CDS” on page 73.
Starting the Application Support server	Chapter 16, “Starting the Application Support Server” on page 89.
Installing new interfaces	Chapter 17, “Using the ASUADMIN Administration Program” on page 97, “Installing and Uninstalling Specific Interfaces” on page 103, and “Installing and Uninstalling Interfaces” on page 120.
Attaching the Application Support server to CICS or IMS	Chapter 17, “Using the ASUADMIN Administration Program” on page 97, “Attaching the Application Support Server” on page 102, or Chapter 18, “Administering the AS Server Using MVS System Commands” on page 115.

Maintaining the Application Support Server

In a typical installation, you would perform the following tasks to maintain the Application Support server:

- Install and uninstall interfaces.

To add a new IDL interface, you must install the server stub. To remove an IDL interface, you must uninstall it, *before* deleting the server stub from your library. To change an IDL interface, you must uninstall the server stub and install the changed one.

- Display operational status.

Monitoring the status of the Application Support server is a common daily task. You can use the administration panels to display information such as the interfaces that are supported by the Application Support server, the state of the Application Support server, and Application Support server definition data. You can also display the operational status by using the MODIFY system command.

- Add, change, or delete users.

Other administration tasks are adding and deleting users of the Application Support server and changing information about these users by using the DCE administrative interface (**dcecp**). These DCE users also need to be mapped to MVS user IDs using an identity mapping file or RACF. If the user is going to perform Application Support server administration tasks, you must change the ACL of the Application Support server to allow the user to perform these tasks.

- Display transactional information (only for AS IMS OTMA or AS CICS servers using transactional RPCs).

You can use the Encina **tkadmin** command from another system or use the RRS panels.

- Stop the attachment of the Application Support server from the CICS or IMS subsystem.

For a variety of reasons, the attachment of the Application Support server to the CICS or IMS subsystem may have to be **stopped**. A typical scenario is moving the Application Support server to another CICS or IMS subsystem. Another example is when client requests have to be **quiesced** (that is, the Application Support server is put into the **quiescent** server state) for throughput reasons. Appendix C, “Server States” on page 169 describes server states. You stop the attachment of the Application Support server from CICS or IMS by using the administration panels or by using the MODIFY system command (see Chapter 18, “Administering the AS Server Using MVS System Commands” on page 115).

- Stop the Application Support server.

Table 4 summarizes the sections in this book that describe these tasks.

Tasks	Book Sections
Installing and uninstalling interfaces	Chapter 17, “Using the ASUADMIN Administration Program” on page 97, “Installing and Uninstalling Specific Interfaces” on page 103, and “Installing and Uninstalling Interfaces” on page 120.
Displaying operational status	“Displaying the Server's Operational Status” on page 117, Chapter 17, “Using the ASUADMIN Administration Program” on page 97, and “Displaying Interfaces and AS Server Operational Data” on page 103.
Adding, changing, or deleting users	Chapter 10, “Creating Administrator and User Accounts” on page 55.
Displaying transactional information (only for AS IMS OTMA or AS CICS servers using transactional RPCs)	Chapter 20, “Using the tkadmin Command with the AS Server” on page 135. (Also see the Administrator's Guide for the Encina Toolkit for your system.)
Stopping the Application Support server attachment	Chapter 17, “Using the ASUADMIN Administration Program” on page 97, “Stopping an Application Support Server Attachment” on page 103, or Chapter 18, “Administering the AS Server Using MVS System Commands” on page 115.
Stopping the Application Support server	Chapter 18, “Administering the AS Server Using MVS System Commands” on page 115.

Installation of Application Support

All files necessary to make the Application Support server run on z/OS are placed in the proper libraries during the installation of Application Support server. The *z/OS Program Directory* describes installation. The sample JCL to start the Application Support server is supplied as members in the SASUSAMP data set:

```

ASPREFIX.SASUSAMP(ASUVSC) - for starting an AS CICS server
ASPREFIX.SASUSAMP(ASUVSA) - for starting an AS IMS server with APPC
ASPREFIX.SASUSAMP(ASUVSI) - for starting an AS IMS server with ISC
ASPREFIX.SASUSAMP(ASUVSO) - for starting an AS IMS server with OTMA

```

Part 2. Configuring the Application Support Server

This part of the book describes the tasks that are required to configure the Application Support server after installing it on the z/OS host system.

Chapter 4. Creating the Environment

Variable File	21
Step 1: Create the Application Support Server Directory	21
Create One Directory for One Application Support Server	21
Create Multiple Directories for Multiple Application Support Servers	22
Step 2: Specify the Directory	22
Step 3: Grant the Application Support Server Access to the Files in the Directory	23
Step 4: Create the Environment Variable File	23
Step 5: Set the Code Page of the Application Support Server	32

Chapter 5. Configuring the AS CICS Server

Requirements for Nontransactional and Transactional RPCs	33
CICS Resource Considerations	33
Steps for Configuring DCE AS CICS	34
Step 1: Define the MRO Connection Path	34
Step 2: Define SESSIONS Attributes	35
Step 3: Define the Transaction to CICS (Optional)	36
Step 4: Set Environment Variables	36
Step 5: Specify CICS Definition Data in CDS	36
Step 6: Configure Encina for Use with Application Support	37
Security Considerations in DCE AS CICS	37
Access to CICS Programs	37
Transactional Considerations	38

Chapter 6. Configuring the AS IMS OTMA Server

IMS Resource Considerations	39
Definitions for DCE AS IMS with OTMA	39
Step 1: Modify the IMS System Definition	40
Step 2: Ensure Environment Variables Are Set	40
Step 3: Configure Encina for Use with Application Support	41
Step 4: Set IMS OTMA Security	41
Operational Considerations When Using AS IMS with OTMA	41

Chapter 7. Configuring the AS IMS APPC Server

Server	43
------------------	----

Definitions for DCE AS IMS with APPC	43
Step 1: Specify APPCPMxx Member in SYS1.PARMLIB	43
Step 2: Define AS IMS to VTAM	44
Step 3: Define AS IMS to RACF	45
Step 4: Ensure Environment Variables for AS IMS with APPC Have Been Set	46
Step 5: Configure the IMS DFSLUEE0 Exit Routine	46

Chapter 8. Configuring the AS IMS ISC Server

Definitions for AS IMS with ISC	47
Step 1: Specify IMS Definitions for TYPE, TERMINAL, and VTAMPOOL	47
Step 2: Specify VTAM Definitions in the VTAM Definition Library	49
Step 3: Define DCE AS IMS with ISC Attributes to the CDS	50
Step 4: Get Access to IMS Programs	50

Chapter 9. Creating the DCE Account for the AS Server

Step 1: Update the Keytab File for the Application Support Server	51
Step 1a: Set the _ASU_KEYTAB_FILE Environment Variable	51
Step 1b: Secure the Keytab File	51
Step 2: Give the AS Server Access to the DCE Endpoint Map	52
Step 3: Select the Group for the Application Support Server	52
Step 4: Create a DCE Account for the Application Support Server	52
Step 5: Ensure Certificate Lifetimes Across Cells	54

Chapter 10. Creating Administrator and User Accounts

Step 1: Register Users with the MVS Security Subsystem	55
Step 2: Create DCE Accounts for Application Support Server Users and Administrators	55
Step 3: Mapping DCE Principal Names to MVS User IDs	56
Mapping DCE Principal Names to MVS User IDs with the Identity Mapping File	57

Modifying and Deleting Identity Mapping Entries	60	Using the ACL Editor to Give Access to the Administration Program	72
Access to the Identity Mapping File	60		
Mapping DCE Principal Names to MVS User IDs with RACF	60		
Chapter 11. Configuring the CDS		Chapter 13. Defining the Application	
Namespace Entries	63	Support Server to CDS	73
Creating CDS Directories for the Application Support Server	63	What Does Defining to CDS Mean?	73
Access to the Application Support Server		ASUADMIN Administration Program	74
Namespace Entries	64	Steps in Defining the Application Support Server to CDS	74
Procedure for Giving Access to the CDS Namespace Entries	64	Entering Application Support Server Attributes	76
If the Application Support Server Definition Has Been Deleted	65		
Creating Authorization Groups for Accessing the CDS Namespace	65	Chapter 14. Example Configuration	
		Worksheets	77
Chapter 12. Giving Access to the		AS CICS Worksheet	77
Administration Program	67	AS IMS with APPC Worksheet	79
Access to the Administration Program	67	AS IMS with OTMA Worksheet	80
ACL Permissions Supported for the Administration Program	68	AS IMS with ISC Worksheet	81
Initial Access Control List	68		
Creating the Initial ACL Database Files	68	Chapter 15. Blank Configuration	
ACL Database File Naming Convention	69	Worksheets	83
Creating the Version File	69	AS CICS Worksheet	83
Creating the ACL Database File	69	AS IMS with APPC Worksheet	85
Example ACL Database File	70	AS IMS with OTMA Worksheet	86
Setting the _EUV_RPC_ACL_FILE Environment Variable	71	AS IMS with ISC Worksheet	87
Switching between Current and Backup Copies of the ACL File	71		
		Chapter 16. Starting the Application	
		Support Server	89
		Startup Procedure for DCE AS CICS	89
		Startup Procedure for DCE AS IMS with OTMA	90
		Startup Procedure for DCE AS IMS with APPC	91
		Startup Procedure for DCE AS IMS with ISC	92

Chapter 4. Creating the Environment Variable File

Each instance of the Application Support server uses environment variables that affect its behavior. Environment variables can be set in the environment variable file, also known as the `envar` file. This file contains declarations of environment variables, one declaration per line, that are specific to z/OS DCE and the Application Support server. You use different environment variables, depending on the type of server you are using (CICS, ISC, APPC, or OTMA) and on other conditions you want to control.

When you start the Application Support server, the `envar` file is read from the specified directory. Because each instance of the Application Support server uses a specific set of environment variables, each Application Support server must have its own `envar` file. Therefore, each Application Support server must have its own directory.

This chapter describes the following steps necessary for creating an environment variable file and making it available to the Application Support server:

- Step 1. Create a directory for the Application Support server.
- Step 2. Specify the directory to the Application Support server.
- Step 3. Grant the Application Support server access to directory files.
- Step 4. Create the `envar` file.
- Step 5. Set the code page of the Application Support server.

This chapter describes these steps.

Important Note: Before you begin, you must obtain an MVS user ID for the Application Support server. This user ID can be the administrator who submitted the job.

Step 1: Create the Application Support Server Directory

You may have only one Application Support server in your distributed computing environment, or you may have several Application Support servers. This section describes the recommended way to set up the directory or directories to contain your environment variable file or files. Each instance of the Application Support server requires different values for some of the Application Support server-specific environment variables. For example, the value of the `_EUV_RPC_ACL_FILE` environment variable that points to the ACL file is unique for each instance of the Application Support server. Thus, each instance of the Application Support server has its own directory and its own `envar` file. That means you have to create one directory for each Application Support server you plan to run. If you have only one instance of the Application Support server, follow the directions in “Create One Directory for One Application Support Server.” If you have more than one instance of the Application Support server, follow the directions in “Create Multiple Directories for Multiple Application Support Servers” on page 22.

Create One Directory for One Application Support Server

Use the `mkdir` command to create a directory for the Application Support server. This directory stores the `envar` file (which is created later—see “Step 4: Create the Environment Variable File” on page 23). For example, to create the directory for the `server_1` Application Support server, enter the following:

```
mkdir /var/AS
mkdir /var/AS/server_1
```

Create Multiple Directories for Multiple Application Support Servers

If you choose to run several instances of the Application Support server under the same user ID, you can create subdirectories under the directory of this common user ID. Each of these subdirectories serves as the directory of each instance of the Application Support server.

Suppose you have three instances of the Application Support server: **server_1**, **server_2**, and **server_3**. To create the directories of the common user ID for the /var/AS Application Support server directory, for example, use the mkdir command to create three subdirectories:

```
mkdir /var/AS
mkdir /var/AS/server_1
mkdir /var/AS/server_2
mkdir /var/AS/server_3
```

Note: You can create the directories of each Application Support server anywhere in the HFS tree, as long as you create one directory for each instance of the Application Support server.

Step 2: Specify the Directory

By default, the environment variable file is read from the user's own directory. Application Support looks in the directory for a file with the name **envar**. If you want to use the directory you created in “Step 1: Create the Application Support Server Directory” on page 21, you can skip this step and continue with “Step 3: Grant the Application Support Server Access to the Files in the Directory” on page 23.

To override the default, use the **_EUV_HOME** environment variable in the startup PROC of the Application Support server to specify a different directory.

For example, given the example startup PROC that follows, the Application Support server uses the envar file /var/AS/server_1/envar:

```

/*****
/* ASUVSC - startup procedure for the DCE to CICS AS server
/*****
/*****
/* Sample JCL to invoke the procedure
/*****
/* //ASUVCICS EXEC ASUVSC,
/* // XSTUBLIB='TSDCEPR.CICS.AUTHLIB',
/* // CICSPFX='SHARE.CICS410',
/* // PARMS=('ENVAR(''_EUV_HOME=/var/AS/server_1''))/'
/* //GO.SYSIN DD *
/* ./AS/server_1
/* p_server_1
/* /*
/*
/* SYSIN contains:
/*          Server CDS name
/*          Server DCE principal name
/*****
//ASUVSC  PROC XSTUBLIB=,REGSIZE=0M,PARMS=,
// OUTCLASS='*',
// CICSPFX='CICS.V4R1'
//GO      EXEC PGM=ASUVSC,REGION=&REGSIZE.,TIME=1440,PARM=&PARMS
/*****
/* LOAD LIBRARIES
/*****
//STEPLIB DD DSN=&XSTUBLIB,DISP=SHR
//          DD DSN=&CICSPFX..SDFHEXCI,DISP=SHR
/*****
//SYSIN   DD DUMMY
//SYSOUT  DD SYSOUT=&OUTCLASS
//SYSPRINT DD SYSOUT=&OUTCLASS
//CEEDUMP DD SYSOUT=&OUTCLASS
//SYSUDUMP DD SYSOUT=&OUTCLASS

```

Figure 3. Example Startup PROC

Step 3: Grant the Application Support Server Access to the Files in the Directory

The MVS user ID under which the Application Support server job runs must be able to change all files in the Application Support server directory and the directory itself. If you choose to create the files under the administrator's ID, you need read and write permission to the files in the directory.

Step 4: Create the Environment Variable File

The environment variable (envar) file must be created in the Application Support server's own directory. The Application Support server product provides example envar files in the following locations:

```

/usr/lpp/dceas/examples/ascics/envar (for AS CICS)
/usr/lpp/dceas/examples/asims/envar (for AS IMS with ISC)
/usr/lpp/dceas/examples/asimso/envar (for AS IMS with OTMA)
/usr/lpp/dceas/examples/asimsa/envar (for AS IMS with APPC)

```

The following four figures show examples for CICS and IMS with various adaptors. Figure 4 on page 24 shows an example envar file for DCE AS CICS.

```
_EUV_SVC_MSG_LOGGING=CONSOLE_LOGGING
_EUV_RPC_ACL_FILE=/var/AS/server_1/as1acl.dat
_EUV_RPC_DYNAMIC_POOL=NO
_ASU_KEYTAB_FILE=/var/AS/server_1/ktabfile
_ASU_IDMAP=/var/AS/server_1/idmap
_ASU_EX_SEC=NO
_ASU_AUTOSTART=NO
_ASU_CICS_CONNECTIONS=20
# The following two variables are needed for transactional RPC support:
# _ASU_TRAN=YES
# _ASU_RESOURCE_MGR_NAME=MVSSYS1CICSSERVER4
ENCINA_CDS_ROOT=./.:/encina
```

Figure 4. Example envar File for DCE AS CICS

Figure 5 shows an example envar file for DCE AS IMS with OTMA.

```
_EUV_SVC_MSG_LOGGING=CONSOLE_LOGGING
_EUV_RPC_ACL_FILE=/var/AS/server_4/as4acl.dat
_ASU_KEYTAB_FILE=/var/AS/server_4/ktabfile
_ASU_IDMAP=/var/AS/server_4/idmap
_ASU_EX_SEC=NO
_ASU_AUTOSTART=NO
_ASU_CONVERSATIONS=64
_ASU_EXPIRE=60
_ASU_XCF_GROUP_NAME=MVSSYS1
_ASU_XCF_SERVER_MEMBER_NAME=SERVER4
_ASU_XCF_IMS_MEMBER_NAME=IMSSYS1
_ASU_RESOURCE_MGR_NAME=MVSSYS1IMSSYS1SERVER4
_ASU_TRAN=YES
ENCINA_CDS_ROOT=./.:/encina
```

Figure 5. Example envar File for DCE AS IMS with OTMA

Figure 6 shows an example envar file for DCE AS IMS with APPC.

```
_EUV_SVC_MSG_LOGGING=CONSOLE_LOGGING
_EUV_RPC_ACL_FILE=/var/AS/server_3/as3acl.dat
_ASU_KEYTAB_FILE=/var/AS/server_3/ktabfile
_ASU_IDMAP=/var/AS/server_3/idmap
_ASU_EX_SEC=NO
_ASU_AUTOSTART=NO
_ASU_SYMBOLIC_NAME=IMS1
_ASU_APF=YES
_ASU_CONVERSATIONS=64
_ASU_EXPIRE=60
_ASU_EXIT=AUTO
_ASU_SERVER_LUNAME=MVSIMS02
```

Figure 6. Example envar File for DCE AS IMS with APPC

Figure 7 on page 25 shows an example envar file for DCE AS IMS with ISC.

```

_EUV_SVC_MSG_LOGGING=CONSOLE_LOGGING
_EUV_RPC_ACL_FILE=/var/AS/server_2/as2acl.dat
_ASU_KEYTAB_FILE=/var/AS/server_2/ktabfile
_ASU_IDMAP=/var/AS/server_2/idmap
_ASU_EX_SEC=NO
_ASU_AUTOSTART=NO

```

Figure 7. Example `envar` File for DCE AS IMS with ISC

Copy the example file to the Application Support server's directory, and then edit the file to suit your requirements. For example, if you are creating an `envar` file for an Application Support server called `server_1`, an DCE AS CICS server, select the example file `/usr/lpp/dceas/examples/ascics/envar` and copy it to your directory, as follows:

```
cp /usr/lpp/dceas/examples/ascics/envar /var/AS/server_1/envar
```

The next two tables divide environment variables into three main types:

- Application Support environment variables
- DCE environment variables

Note: This is not a complete list of all environment variables that you can specify for z/OS DCE. This list includes only environment variables that affect the Application Support server. For a complete list of DCE environment variables, see the *z/OS DCE Administration Guide*.

- Encina environment variables.

Within each of these subsections, the environment variables are alphabetized. The following table indicates which environment variables to use for CICS, APPC, OTMA, and ISC.

<i>Table 5 (Page 1 of 2). Relevant Environment Variables</i>				
Environment Variable Name	APPC	OTMA	ISC	CICS
Application Support Environment Variables				
_ASU_APF	X			
_ASU_AUTOSTART	X	X	X	X
_ASU_CICS_CONNECTIONS				X
_ASU_CONVERSATIONS	X	X		
_ASU_EX_SEC	X	X	X	X
_ASU_EXIT	X			
_ASU_EXPIRE	X	X		
_ASU_IDMAP	X	X	X	X
_ASU_KEYTAB_FILE	X	X	X	X
_ASU_RESOURCE_MGR_NAME		X ²		X ²
_ASU_SERVER_LUNAME	X			
_ASU_SYMBOLIC_NAME	X			
_ASU_TRAN		X ²		X ²
_ASU_XCF_GROUP_NAME		X		
_ASU_XCF_IMS_MEMBER_NAME		X		
_ASU_XCF_SERVER_MEMBER_NAME		X		

² You need this only for transactional RPCs.

<i>Table 5 (Page 2 of 2). Relevant Environment Variables</i>				
Environment Variable Name	APPC	OTMA	ISC	CICS
SVC_VEN_DBG	X	X	X	X
DCE Environment Variables				
_EUV_ENVAR_FILE	X	X	X	X
_EUV_HOME	X	X	X	X
_EUV_LDR_MAX_XSTUBS	X	X	X	X
_EUV_RPC_ACL_FILE	X	X	X	X
_EUV_RPC_COMM_TIMEOUT	X	X	X	X
_EUV_RPC_DYNAMIC_POOL				X
_EUV_SEC_KRB5CCNAME_FILE	X	X	X	X
_EUV_SVC_MSG_LEVEL	X	X	X	X
_EUV_SVC_MSG_LOGGING	X	X	X	X
NLSPATH	X	X	X	X
Encina Environment Variables				
ENCINA_CDS_ROOT		X ²		X ²
ENCINA_TRACE		X ²		X ²
ENCINA_TRACE_VERBOSE		X ²		X ²

Table 6 describes in detail each of the Application Support server-specific environment variables that affect the Application Support server.

<i>Table 6 (Page 1 of 7). Environment Variables for Application Support</i>	
Name	Description
Application Support Environment Variables	
_ASU_APF For APPC only.	<p>Specifies whether the AS IMS server module runs from an APF-authorized library and can use privileged functions. The format is:</p> <p><code>_ASU_APF=yesno</code></p> <p>Valid values of <i>yesno</i> are:</p> <p>YES The AS IMS server can assume the MVS user ID identity associated with the DCE principal that generated an incoming RPC request when calling an IMS transaction. If you specify <i>yes</i> and the AS IMS server module was not run from an APF-authorized library, the server stops during its initialization phase.</p> <p>If you do not specify the <code>_ASU_APF</code> environment variable or if you specify a value that is not valid, the default is YES.</p> <p>NO All IMS transactions are called under the MVS user ID identity of the AS IMS server.</p>

Table 6 (Page 2 of 7). Environment Variables for Application Support

Name	Description
<p>_ASU_AUTOSTART For all servers.</p>	<p>Specifies whether the Application Support server automatically issues the AS STARTATT command when the Application Support server is started. _ASU_AUTOSTART controls only the initial invocation of the STARTATT command. When the Application Support server is operational, you can manually issue STOPATT and STARTATT commands through the MVS MODIFY system command or from the administration panels. The format is:</p> <p><code>_ASU_AUTOSTART=yesno</code></p> <p>Valid values of <i>yesno</i> are:</p> <p>YES This indicates issuing the STARTATT command automatically whenever the Application Support server is started.</p> <p>NO This indicates that the STARTATT command is not issued automatically. It must be issued manually with the MVS MODIFY command or by selecting the appropriate option from the Application Support Primary Options ISPF administration panel.</p> <p>If you do not specify the _ASU_AUTOSTART environment variable or if you specify a value that is not valid, the default is NO.</p>
<p>_ASU_CICS_CONNECTIONS For CICS only.</p>	<p>Sets the number of executor threads that are initiated when the Application Support server is initiated during the Start Attachment processing. The format is:</p> <p><code>_ASU_CICS_CONNECTIONS=nn</code></p> <p>Valid values of <i>nn</i> are integers from 10 to 25 indicating the number of threads to be initiated.</p> <p>When the Application Support server is attached to CICS, the number of executor threads is increased by the number specified for _ASU_CICS_CONNECTIONS to handle subsequent transaction requests.</p> <p>This environment variable can be set by adding an entry in the envvar file in the Application Support server's home directory. For example, if there are 20 executor threads, you would make the following declaration:</p> <p><code>_ASU_CICS_CONNECTIONS=20</code></p> <p>Note: You must set the value of the RECEIVECOUNT parameter in the DEFINE SESSIONS control statement of the CICS definition of the Application Support server to be greater than or equal to the value of the _ASU_CICS_CONNECTIONS environment variable.</p>
<p>_ASU_CONVERSATIONS For APPC or OTMA.</p>	<p>Specifies the number of simultaneous conversations that can be active between the AS IMS server and a partner IMS system at one time. The format is:</p> <p><code>_ASU_CONVERSATIONS=nn</code></p> <p>Valid values of <i>nn</i> are integers from 10 to 10000. If you do not specify the _ASU_CONVERSATIONS environment variable or if you specify a value is not valid, the default is 64. It is recommended that you set _ASU_CONVERSATIONS to be less than or equal to the DSESLIM parameter of the VTAM APPL definition statement.</p>

Table 6 (Page 3 of 7). Environment Variables for Application Support

Name	Description
<p>_ASU_EX_SEC For all servers.</p>	<p>Specifies whether you want to use RACF to map DCE user IDs to MVS user IDs. The format is:</p> <p><code>_ASU_EX_SEC=yesno</code></p> <p>The valid values of <i>yesno</i> are:</p> <p>YES indicates that the Application Support server should use RACF to perform identity mapping. <code>_ASU_EX_SEC</code> must be set to YES in order to use RACF. If <code>_ASU_EX_SEC</code> is set to YES and the level of RACF that contains the <code>R_dceruid</code> function is not available, the server stops.</p> <p>NO indicates using the identity mapping table. If NO is specified, the <code>_ASU_IDMAP</code> variable must be set to a valid mapping file. See “Step 3B: Set the <code>_ASU_IDMAP</code> Environment Variable” on page 59 for information on setting <code>_ASU_IDMAP</code>.</p> <p>If you do not specify the <code>_ASU_EX_SEC</code> environment variable or if you specify a value that is not valid, the default is NO.</p>
<p>_ASU_EXIT For APPC only.</p>	<p>Informs the AS IMS with APPC server about the presence of the AS IMS-supplied IMS DFSLUEEO exit routine. The format is:</p> <p><code>_ASU_EXIT=autoyesno</code></p> <p>Valid values of <i>autoyesno</i> are:</p> <p>AUTO The AS IMS server automatically determines if the IMS DFSLUEEO exit is installed by sending a command that is not valid to IMS and interpreting the return information. IMS must be configured to allow commands through APPC from the AS IMS server for AUTO to work correctly. If you specify AUTO and the AS IMS server cannot issue IMS commands, STARTATT fails.</p> <p>If you do not specify the <code>_ASU_EXIT</code> environment variable or if you specify a value that is not valid, AUTO is the default.</p> <p>YES The AS IMS server assumes that the exit is installed.</p> <p>NO The AS IMS server assumes that the exit is not installed.</p> <p>If this variable is set to YES and the exit is <i>not</i> installed or if it is set to NO and the exit <i>is</i> installed, incorrect handling of RPC input and output data may occur, with unpredictable results.</p>
<p>_ASU_EXPIRE For APPC and OTMA only.</p>	<p>Specifies the approximate time in seconds that the AS IMS server holds an orphaned conversation before deallocating it. An orphaned conversation can occur because the DCE client program or APPC/IMS or OTMA has stopped communicating with the AS IMS server or IMS is not responding.</p> <p>The format is:</p> <p><code>_ASU_EXPIRE=nn</code></p> <p>Valid values of <i>nn</i> are integers from 10 to 3600, indicating the number of seconds.</p> <p>If you do not specify the <code>ASU_EXPIRE</code> environment variable or if you specify a value that is not valid, the default is 60.</p>

Table 6 (Page 4 of 7). Environment Variables for Application Support

Name	Description
<p>_ASU_IDMAP For all servers.</p>	<p>Sets the pathname of the Identity Mapping File. This variable must be present if _ASU_EX_SEC is omitted or set to NO. Use this environment variable to map a DCE principal name to an MVS user ID in an identity mapping file. The format is:</p> <p><code>_ASU_IDMAP=<i>hfspathname</i></code></p> <p>The <i>hfspathname</i> is the pathname of the Identity Mapping file. For example, if the pathname of the Identity Mapping file is <code>/var/AS/server_1/idmap</code>, make the following entry in the <code>envar</code> file:</p> <p><code>_ASU_IDMAP=/var/AS/server_1/idmap</code></p>
<p>_ASU_KEYTAB_FILE For all servers.</p>	<p>Specifies the pathname of the keytab file. The format is:</p> <p><code>_ASU_KEYTAB_FILE=<i>keytabpathname</i></code></p> <p>The <i>keytabpathname</i> is the pathname of the keytab file for the Application Support server to use. For example, if the pathname of the keytab file is <code>/var/AS/server_1/ktabfile</code>, make the following entry in the <code>envar</code> file:</p> <p><code>_ASU_KEYTAB_FILE=/var/AS/server_1/ktabfile</code></p> <p>If you do not specify the _ASU_KEYTAB_FILE environment variable, the Application Support server uses the default keytab file of /krb5/v5srvtab when it is started.</p>
<p>_ASU_RESOURCE_MGR_NAME For OTMA and CICS only.</p>	<p>For transactional RPCs, defines a sysplex-unique name that the AS server uses when registering with the RRS component. The format is:</p> <p><code>_ASU_RESOURCE_MGR_NAME=<i>uniquename</i></code></p> <p>The <i>uniquename</i> consists of 1 to 32 uppercase alphanumeric or national (#, \$, @) characters.</p> <p>The Application Support server does not check the contents of this variable for valid characters. It accepts up to 32 characters of information.</p> <p>If you do not specify _ASU_RESOURCE_MGR_NAME and if _ASU_TRAN has a value of YES or it defaults to YES the AS CICS or AS IMS server stops during its initialization phase. There is no default value for _ASU_RESOURCE_MGR_NAME.</p> <p>After you start the AS CICS or AS IMS server with a specific <i>uniquename</i>, it must continue to use this <i>uniquename</i> on subsequent invocations.</p>
<p>_ASU_SERVER_LUNAME For APPC only.</p>	<p>Specifies the logical unit name associated with the AS IMS server. The format is:</p> <p><code>_ASU_SERVER_LUNAME=<i>luname</i></code></p> <p>The value specified for <i>luname</i> is the name specified on the LUADD statement for the AS IMS server. See Chapter 7, "Configuring the AS IMS APPC Server" on page 43 for more information on the LUADD statement. If you do not specify this environment variable, the AS IMS server stops during its initialization phase. There is no default value for _ASU_SERVER_LUNAME.</p>

Table 6 (Page 5 of 7). Environment Variables for Application Support

Name	Description
<p><code>_ASU_SYMBOLIC_NAME</code> For APPC only.</p>	<p>Specifies the symbolic destination name of the partner IMS system. The format is:</p> <p><code>_ASU_SYMBOLIC_NAME=symbname</code></p> <p>The value specified for <i>symbname</i> is the name specified on the DESTNAME keyword in the APPC/MVS side information file. (See Chapter 7, “Configuring the AS IMS APPC Server” on page 43 for more information on the DESTNAME statement.) This allows multiple AS IMS servers to be started, each communicating with the same or different IMS system. If you do not specify this environment variable, the AS IMS server stops during its initialization phase. There is no default value for <code>_ASU_SYMBOLIC_NAME</code>.</p>
<p><code>_ASU_TRAN</code> For OTMA and CICS only.</p>	<p>Specifies whether the server accepts transactional RPCs. The format is:</p> <p><code>_ASU_TRAN=yesno</code></p> <p>The valid values of yesno are:</p> <p>YES The server is capable of accepting a transactional RPC. All the transactional prerequisites must be met. See “Before You Begin Configuring Your Application Support Server” on page 9 for information about transactional prerequisites.</p> <p>NO The server cannot accept a transactional RPC.</p> <p>If you do not specify the <code>_ASU_TRAN</code> environment variable or if you specify a value that is not valid, the default is YES for AS IMS OTMA. It is NO for AS CICS.</p>
<p><code>_ASU_XCF_GROUP_NAME</code> For OTMA only.</p>	<p>Defines the XCF group name that both the AS IMS server and the IMS/ESA system use. This name must match the name on the GRPNAME= parameter of the IMS procedure. (See page 40.) The format is:</p> <p><code>_ASU_XCF_GROUP_NAME=groupname</code></p> <p>The <i>groupname</i> consists of 1 to 8 alphanumeric or national (#, \$, @) characters.</p> <p>The AS server does not check the contents of this variable for valid characters. It accepts up to eight characters of information, translating lowercase alphabetic characters to uppercase.</p> <p>If you do not specify this environment variable, the AS IMS server stops during its initialization phase. There is no default value for <code>_ASU_XCF_GROUP_NAME</code>.</p>
<p><code>_ASU_XCF_IMS_MEMBER_NAME</code> For OTMA only.</p>	<p>Defines the XCF member name that the IMS/ESA system uses. This name must be the same as the IMS APPLID. The format is:</p> <p><code>_ASU_XCF_IMS_MEMBER_NAME=membername</code></p> <p>The <i>membername</i> consists of 1 to 16 alphanumeric or national (#, \$, @) characters.</p> <p>The AS server does not check the contents of this variable for valid characters. It accepts up to 16 characters of information, translating lowercase alphabetic characters to uppercase.</p> <p>If you do not specify this environment variable, the AS IMS server stops during its initialization phase. There is no default value for <code>_ASU_XCF_IMS_MEMBER_NAME</code>.</p>

Table 6 (Page 6 of 7). Environment Variables for Application Support

Name	Description
<p><code>_ASU_XCF_SERVER_MEMBER_NAME</code> For OTMA only.</p>	<p>Defines the XCF member name that the AS IMS server uses. The format is:</p> <p><code>_ASU_XCF_SERVER_MEMBER_NAME=membername</code></p> <p>The <i>membername</i> consists of 1 to 16 alphanumeric or national (#, \$, @) characters. Each AS IMS server in a sysplex must specify an XCF member name that is unique within the XCF group that contains both the AS and IMS servers.</p> <p>The AS server does not check the contents of this variable for valid characters. It accepts up to 16 characters of information, translating lowercase alphabetic characters to uppercase.</p> <p>If you do not specify this environment variable, the AS IMS server stops during its initialization phase. There is no default value for <code>_ASU_XCF_SERVER_MEMBER_NAME</code>.</p>
<p><code>SVC_VEN_DBG</code> For all servers.</p>	<p>Activates the Application Support server internal debug trace (for use by IBM service personnel only).</p>
DCE Environment Variables Application Support Uses	
<p><code>_EUV_ENVAR_FILE</code> For all servers.</p>	<p>Specifies the pathname of the environment variable file for the Application Support server to use. The default value is \$HOME/envar.</p>
<p><code>_EUV_HOME</code> For all servers.</p>	<p>Specifies the home directory of the Application Support server. This is specified in the startup PROC of the Application Support server, not in the envar file. For more information on <code>_EUV_HOME</code>, see “Step 2: Specify the Directory” on page 22.</p>
<p><code>_EUV_LDR_MAX_XSTUBS</code> For all servers.</p>	<p>Controls the maximum number of stubs (interfaces) that can simultaneously be in memory during the operation of the Application Support server. If not specified, the default is 500.</p>
<p><code>_EUV_RPC_ACL_FILE</code> For all servers.</p>	<p>Sets the pathname of the default ACL file. Setting this variable is described in Chapter 12, “Giving Access to the Administration Program” on page 67.</p>
<p><code>_EUV_RPC_COMM_TIMEOUT</code> For all servers.</p>	<p>Overrides the communication timeout default value. The timeout value can be any integer from 0 to 10, where each integer stands for a length of time, usually in seconds, for the attempt to communicate. 0 attempts to communicate for 1 second. 5, the default, attempts to communicate for 30 seconds. 10, which is not recommended for the Application Support environment, attempts to communicate infinitely.</p>
<p><code>_EUV_RPC_DYNAMIC_POOL</code> For CICS only.</p>	<p>Specifies whether a dynamic pool of executor threads is created. This must be set to NO, meaning that the static pool is used. The example environment variable file sets this variable to NO.</p>
<p><code>_EUV_SEC_KRB5CCNAME_FILE</code> For all servers.</p>	<p>Specifies the file that contains the KRB5CCNAME environment variable. The KRB5CCNAME environment variable contains the name of the Security credentials cache file.</p>
<p><code>_EUV_SVC_MSG_LEVEL</code> For all servers.</p>	<p>Sets the minimum level of error severity at which messages are displayed. Chapter 22, “Problem Determination” on page 155 describes setting this variable, and “Setting <code>_EUV_SVC_MSG_LEVEL</code>” on page 155 provides additional information. <i>z/OS DCE Messages and Codes</i> also discusses it in more detail.</p>
<p><code>_EUV_SVC_MSG_LOGGING</code> For all servers.</p>	<p>Sets where the messages are displayed. Chapter 22, “Problem Determination” on page 155 describes setting this variable, and “Setting <code>_EUV_SVC_MSG_LOGGING</code>” on page 156 provides additional information. <i>z/OS DCE Messages and Codes</i> discusses it in more detail.</p>

Table 6 (Page 7 of 7). Environment Variables for Application Support

Name	Description
NLSPATH For all servers.	Specifies the search path for message catalogs. This environment variable is optional if only messages in the English language are required. The format is: NLSPATH= <i>searchpath</i> The <i>searchpath</i> is the pathname of the message file for the Application Support server to use. The default appears in the following. (The %L is the LANG environment variable which has a default value of En_US, and %N is the catalog name.) /usr/lib/nls/msg/En_US.IBM-1047/%N: /usr/lib/nls/msg/%L/%N: /usr/lib/nls/msg/prime/%N:
Encina Environment Variables Application Support Uses	
ENCINA_CDS_ROOT OTMA and CICS only.	You need to specify this if you wish to use a root directory name other than the default of ./encina.
ENCINA_TRACE OTMA and CICS only.	Activates the Encina internal debug trace (which is for use only for IBM service personnel). ENCINA_TRACE=all:0x1f
ENCINA_TRACE_VERBOSE OTMA and CICS only.	Activates the Encina internal debug trace (which is for use only for IBM service personnel). ENCINA_TRACE_VERBOSE=2

Step 5: Set the Code Page of the Application Support Server

The **locale** setting of the Application Support server determines how character strings are handled. Ensure that the code page component of the locale setting for the Application Support server matches the code page of the CICS or IMS subsystem. This enables variant characters to be used in (CICS or IMS) transaction calls. You can set the code page of the Application Support server by using the LC_ALL environment variable (in the envvar file of the Application Support server). For more information on environment variables for setting code pages, see the *z/OS C/C++ Programming Guide*.

If the code page of the Application Support server is set to a value that is different from the base DCE services, variant characters may not be interpreted correctly. As such, CDS names (for example, Application Support server names) must not contain variant characters such as square brackets ([]), number sign (#), or at sign (@).

Chapter 5. Configuring the AS CICS Server

This chapter describes the CICS resource definitions that are necessary to enable the interaction between the Application Support server and the CICS region that it supports. Definition of these resources can be done through the Resource Definition Online (RDO) facility or through DFHCSDUP commands.

In the following sections that describe these definitions, specific values for each resource attribute are listed. If an attribute is not described, it is either not relevant to DCE AS CICS or it may assume its default value.

Example control statements that can be used as input to DFHCSDUP are also presented at the end of each section.

The last part of this section describes the security considerations in controlling access to the callable CICS application programs.

Requirements for Nontransactional and Transactional RPCs

You can configure a CICS server two ways:

1. For nontransactional RPCs
2. For transactional RPCs.

For nontransactional RPC support, you need CICS/ESA 4.1 or later. For transactional RPC support, you need:

- CICS Transaction Server 1.3 or later
- The z/OS Encina Toolkit
- RRS
- The MVS System Logger Component

For information about RRS logging, see *z/OS MVS Programming: Resource Recovery*. For more information about CICS logging, see the *CICS Transaction Server for OS/390 Installation Guide*. For information about requirements for the MVS System Logger Component, see the System Logger chapter in the *z/OS MVS Programming: Assembler Services Guide*.

Note: The Application Support server, CICS, and RRS must be on the same system.

CICS Resource Considerations

When a CICS transaction program runs as part of a transactional RPC using AS CICS, that transaction is active until the distributed commitment process is complete. Any resources that the CICS transactional program uses, including sessions to other CICS regions and locks, are also active during this period. Locks acquired during an RPC request may be held until the synchronization point (when resources are synchronized).

Steps for Configuring DCE AS CICS

The following are the steps you must follow to configure DCE AS CICS. Before beginning these steps, you must configure CICS. For transactional RPCs, you must set the CICS system initialization parameter RRMS to YES, either in the System Initialization Table (SIT) or on the SIT override parameters.

Step 1. Define the MRO connection path

Step 2. Define SESSIONS attributes

Step 3. Define the transaction to CICS (optional)

Step 4. Set the environment variables

Step 5. Specify CICS definition data in CDS

Step 6. If you are using transactional RPCs, configure Encina for use with Application Support

This chapter also contains security information for DCE AS CICS in “Security Considerations in DCE AS CICS” on page 37. See also “Transactional Considerations” on page 38.

Step 1: Define the MRO Connection Path

You must define the Multi-region Operation (MRO) communication path from the Application Support server to the attached CICS region. Use a CONNECTION definition to do this. The values of the relevant attributes and an explanation for each are listed in Table 7.

Attribute	Value	Explanation
ACCESSMETHOD	IRC	Communication between the CICS system and the named Application Support server is accomplished through the interregion communication (IRC) program.
ATTACHSEC	IDENTIFY	Requests for application execution within the CICS system are tagged with the MVS user ID of the client making the request. IDENTIFY is required and sufficient for access control purposes because the Application Support server authenticates the identity of the client before sending the request to the CICS system.
CONNECTION	<i>anyname</i>	<i>anyname</i> is a 4-character nickname for the Application Support server being defined. This nickname must be unique within all other CONNECTION nicknames known to the same CICS system. The Application Support server does not use or specifically refer to this name.
CONNTYPE	SPECIFIC	This defines that the connection is for communication from a non-CICS client program to the CICS region, and is specific. A specific connection is an MRO link with one or more sessions dedicated to a single user in a client program.
NETNAME	<i>Server-generic-applid</i>	This is the RDO netname that corresponds to the <i>Server Generic Appl-ID</i> of the defined Application Support server. The CICS system uses this name to identify the attached Application Support server. The <i>Server-generic applid</i> is assigned when the Application Support server is defined to CDS through the Application Support server administration panels. For more information on defining the Application Support server to CDS, see Chapter 13, “Defining the Application Support Server to CDS” on page 73. For information on choosing server names, see “Application ID Naming Convention” on page 35.
PROTOCOL	EXCI	EXCI is the protocol option.

Following are example control statements that can be used as input to the DFHCSDUP utility to generate a CONNECTION definition. In the example, a value of AS1 has been supplied for the CONNECTION attribute and a value of CICSAS1 has been supplied for the NETNAME attribute.

```
DEFINE CONNECTION(AS1)
  CONNTYPE(SPECIFIC)
  GROUP(AS1GRP) AC(IRC) AT(IDENTIFY)
  DE(CONNECTION DEFINITION FOR APPLICATION SUPPORT SERVER AS1)
  PROTOCOL(EXCI)
  INS(YES) NETNAME(CICSAS1)
```

Application ID Naming Convention: The CICS and Server Generic Application IDs may contain up to eight characters and must follow Assembler language rules. These Application IDs must start with an alphabetic character. The acceptable characters are A – Z, 0 – 9, \$, @, and #. Lowercase characters are converted to uppercase.

Step 2: Define SESSIONS Attributes

The values of the relevant SESSIONS attributes and explanation for each are listed in Table 8.

<i>Table 8. SESSIONS Attributes and Values</i>		
Attribute	Value	Explanation
CONNECTION	<i>anyname</i>	This is any 4-character nickname for the Application Support server being defined, as specified in the CONNECTION definition. This nickname links the SESSIONS definition with the associated CONNECTION definition. The Application Support server does not use or specifically refer to this name.
PROTOCOL	EXCI	EXCI must be specified for an MRO connection. If you specify EXCI, you must leave the SENDCOUNT attribute blank (its default value).
RECEIVECOUNT	<i>number1</i>	The number of parallel sessions from the named Application Support server. CICS uses the number specified to generate the last characters of the TCTTE names used to communicate with the named Application Support server. The value of RECEIVECOUNT must then be used as the basis to set the value of the _ASU_CICS_CONNECTIONS environment variable. Note: The value of RECEIVECOUNT must be greater than or equal to the value of the _ASU_CICS_CONNECTIONS environment variable. See “Step 4: Set Environment Variables” on page 36 for more information on setting this variable.
SESSIONS	<i>sessionname</i>	This name identifies the SESSIONS definition on the CSD. The active CICS system does not use it, and the attached Application Support server does not use or refer to it.

Following are example control statements that can be used as input to the DFHCSDUP utility to generate a SESSIONS definition. In the example, a value of AS1 has been supplied for the CONNECTION attribute, a value of SESSAS1 has been supplied for the SESSIONS attribute, and a value of 20 has been supplied for the RECEIVECOUNT attribute.

```
DEFINE SESSIONS(SESSAS1)
  GROUP(AS1GRP) CONNECTION(AS1)
  DESCRIPTION(SESSIONS DEFINITION FOR AS1 CONNECTION)
  PROTOCOL(EXCI)
  RECEIVECOUNT(20) RECEIVEPFX(S1)
```

Step 3: Define the Transaction to CICS (Optional)

You can define the callable CICS transactions using **DFHMIRS** as the Program Name. For example:

```
DEFINE TRANSACTION(MIN)
  GROUP(ASIGRP)
  DESCRIPTION(DEFINITION FOR MINMAX)
  PROGRAM(DFHMIRS)
```

In this example, the transaction ID **MIN** specifies the transaction identifier defined in the IDL file.

Note: For transactional RPCs, the transaction ID must be the same for all CICS transactions that are called within one Encina transaction or one unit of work.

For more details on DFHMIRS, see the *CICS Resource Definition Guide*.

Step 4: Set Environment Variables

An AS CICS server receives configuration information from its SYSIN DD input stream, which supplies the server's CDS directory and its DCE principal name, and from its environment variables. Ensure that you set the environment variables listed in the example worksheet (see Table 9 on page 77) in the `envvar` file (or that the default values, if available, are the values you want).

The `_ASU_CICS_CONNECTIONS` environment variable determines the number of executor threads initiated during the **Start Attachment** processing. The Application Support server initially starts with five executor threads for management calls. When the Application Support server is attached to CICS, the number of executor threads increases by the number specified as the value of the `_ASU_CICS_CONNECTIONS` environment variable to handle subsequent transaction requests. Be sure that this environment variable is set in the `envvar` file.

If you are using transactional RPCs, you need to set the `_ASU_RESOURCE_MGR_NAME` environment variable to *uniquename* and set the `_ASU_TRAN` environment variable to YES. This identifies your Application Support server to RRS.

See “Step 4: Create the Environment Variable File” on page 23 and *z/OS DCE Administration Guide* for more information about environment variables. For more information about CICS, see:

- *CICS Supplied Transactions*
- *CICS System Programming Reference*
- *CICS Recovery and Restart Guide*

Step 5: Specify CICS Definition Data in CDS

When the Application Support server is defined to CDS, CICS data is used. Chapter 13, “Defining the Application Support Server to CDS” on page 73 describes this.

Keep the following information for later use:

- APPLID defined for the CICS subsystem.

In the Create a CICS Application Support Server Definition panel, this must match the value specified in the **CICS Generic Application ID** field.

- NETNAME parameter of the CONNECTION definition.

In the Create a CICS Application Support Server Definition panel, this must match the value specified in the **Server Generic Application ID** field.

Step 6: Configure Encina for Use with Application Support

If you are using transactional RPCs, you also need to configure Encina for use with Application Support. The Encina components require the use of a CDS directory structure. It is recommended that you use the default Encina root directory name `./encina`. You also need a directory entry called `trpc` in that directory. For example, if you use the default root directory, this directory entry is called `./encina/trpc`. If these directory entries do not exist in a cell, you need to create them. See the *z/OS Encina Toolkit Executive Guide and Reference* for information about creating `./encina` and `./encina/trpc`.

If you want to change the default root directory name, you need to set the environment variable `ENCINA_CDS_ROOT` to the name you want to use. For more information about using an Encina directory other than the default, see the Encina Toolkit administrator's documentation for your system.

The AS server principal needs write access to the `./encina` directory and write and insert access to the `./encina/trpc` directory. You are recommended to create a DCE group, for example, `encina-servers`, add Application Support to that group, and give that group write authority to the directories. General read access must be given because principals of other servers may need to read entries in these directories.

Security Considerations in DCE AS CICS

Appropriate security measures control access to the callable CICS application programs. The following sections describe the security considerations for these CICS transactions.

Access to CICS Programs

A request for program execution within a CICS system typically occurs as a result of a transaction being initiated (from a terminal or a `START` command) or as a result of a `LINK/XCTL` request from a program already running under a transaction. In DCE AS CICS, the client application program does not run in a CICS environment. Instead, the client application program requests execution of a CICS application by program name, as is the case with `EXEC CICS LINK`, but is not associated with a transaction known to CICS.

The attached transaction defines the environment under which a CICS program runs. This includes, but is not limited to, the defined security controls to be enforced. Therefore, DCE AS CICS associates a transaction identifier with the request for CICS program execution at the time the Application Support server receives the request. The Application Support server obtains the appropriate transaction identifier and program identifier from the server stub created from the compiled Interface Definition Language (IDL) file. The application programmer defines the RPC interface and its associated data types in the IDL file.

Access Control on Target CICS Applications: It is important to note the significance of the operational data contained in the IDL statements that define the relationship between an operation within an RPC interface known to the Application Support server and the associated transaction and application program to run in the attached CICS region.

To avoid the possibility of circumventing defined access control against a target CICS application, an authorized system administrator must specify the transaction identifier and associated CICS program name in the IDL statements comprising the RPC interface definition. This administration task, necessary for supporting DCE AS CICS, is in some respects analogous to defining the initial program in the `TRANSACTION` resource definition. It allows existing CICS security mechanisms to be applied to requests made through the Application Support server.

It is recommended that only an authorized user be in charge of compiling all IDL definitions for the interfaces and installing the load modules in the stub library.

Authorizing Mechanisms in CICS: Two mechanisms exist within a CICS system for authorizing access to CICS application programs. Transaction-attach security can be employed to restrict or permit specific user access to transactions, thereby providing access control against the programs (and all other resources) that run under the transactions. Another form of security within the CICS system is resource security. Resource security can govern access to specific resources a transaction uses. One such resource is the application program itself. At the discretion of the administrator, these forms of security may or may not be employed to govern access to CICS applications attempted by client applications through DCE AS CICS.

Specifying the SEC=YES and XTRAN=YES (or XTRAN=equivalent-installation-specified-classname) system initialization table (SIT) options enables transaction-attach security. Every time a transaction is initiated, including transactions associated with CICS programs called through the Application Support server, CICS issues an authorization request to determine whether the user associated with the request is authorized for that transaction. This security mechanism requires that the appropriate transaction profiles be RACF (or equivalent) defined to resource class GCICSTRN, TCICSTRN, or an appropriate installation-defined resource class.

Typically, access to the program associated with the transaction can be controlled through the use of transaction-attach security as described in the preceding. However, additional security may be required to govern access to programs at a finer granularity. Application program security is enabled when the SEC=YES and XPPT=YES (or XPPT=equivalent installation-defined-classname) system initialization table (SIT) options are specified, and RESSEC=YES is specified in the TRANSACTION resource definition of the transaction under which the program runs. This security mechanism requires that the appropriate program profiles be RACF (or equivalent) defined to resource class MCICSPPT, NCICSPPT, or an appropriate installation-defined resource class. For more information, see the *CICS RACF Security Guide* and *CICS External Interfaces Guide*.

The methods for mapping user IDs (which “Step 3: Mapping DCE Principal Names to MVS User IDs” on page 56 describes) map DCE principal names to their corresponding MVS user IDs. The Application Support server uses the Identity Mapping file or RACF to obtain the DCE principal's equivalent MVS user ID. The CICS security mechanism described in this section controls access to the CICS resources based on the equivalent MVS user ID of the DCE client.

Transactional Considerations

- RRS must be started before the AS CICS server is started. If RRS is stopped and restarted while the AS CICS server is running, you must stop and restart the AS CICS server.
- When you enable a CICS adaptor to accept transactional RPCs, the AS CICS server module and stub library must run APF-authorized.
- You can use Encina **tkadmin** commands to determine the state of the server and Encina transactions. See Chapter 20, “Using the tkadmin Command with the AS Server” on page 135 and the Administrator's Guide for the Encina Toolkit for your system for more information.

Chapter 6. Configuring the AS IMS OTMA Server

This is the first of three chapters about methods of communication between the Application Support server and IMS:

- OTMA
- APPC
- ISC

You should choose one of these methods and stay with it because changing methods generally requires programming changes. For information about deciding which adaptor to use, see Table 1 on page 9.

This chapter describes the use of OTMA. There are two ways to configure an OTMA server. First, you must determine whether your installation plans to use transactional RPCs (see “For Transactional RPCs” on page 4 for background information). The `_ASU_TRAN` environment variable specifies this.

If you are using the OTMA adaptor, you need:

- IMS/ESA Version 6.1 or later
- The MVS Cross System Coupling Facility

For transactional RPCs, you additionally need:

- The z/OS Encina Toolkit Executive
- RRS
- The MVS System Logger Component (RRS uses this).

For information about requirements for the MVS System Logger Component, see the System Logger chapter in the *z/OS MVS Programming: Assembler Services Guide*.

Note: The Application Support server, IMS, and RRS must be on the same system.

IMS Resource Considerations

When an IMS transaction program runs as part of a transactional RPC using AS IMS, IMS considers that transaction to be active until the distributed commitment process is complete. Therefore, any resources that the IMS transactional program uses, including IMS message regions and locks, are also active during this period. If within the scope of a single Encina transaction multiple IMS transaction programs are called (or the same transaction program is called multiple times), there must be enough message regions available for each of these IMS transaction programs to be active concurrently.

If you are grouping IMS transactions into Encina transactions, you may wish to code `SCHDTYP=PARALLEL` and specify `PARLIM=0` on your transaction definitions.

Definitions for DCE AS IMS with OTMA

This chapter takes you through the steps you need to perform to configure IMS for the Application Support server when you are using the OTMA communications method. Before beginning these steps, make sure RRS is configured and operational. (See the *z/OS MVS Programming: Resource Recovery* for details.)

Step 1. Modify the IMS system definition

Step 2. Check settings of AS IMS with OTMA environment variables

- Step 3. Configure Encina for use with Application Support
- Step 4. Set up your IMS OTMA security profile.

For detailed information about OTMA, see the *IMS/ESA Open Transaction Manager Access Guide*.

Step 1: Modify the IMS System Definition

To use OTMA, you do not need to make any specific definitions in IMS to connect to an Application Support server. However, you do need to enable OTMA itself in IMS. During system definition, in the IMS procedure, specify:

OTMA=YES

and

GRNAME=*name*

where *name* is the name of the XCF group that IMS should join.

You also need to specify this group name on the `_ASU_XCF_GROUP_NAME` environment variable. See page 30. (You may want to take a moment to write this value on the allocated line in the blank OTMA worksheet on page 86.)

The IMS APPLID is the XCF member name that IMS uses when it communicates through OTMA. You store this in the `_ASU_XCF_IMS_MEMBER_NAME` environment variable.

Step 2: Ensure Environment Variables Are Set

An AS IMS OTMA server receives configuration information from two sources:

- Its SYSIN DD input stream
- Its environment variables

Just as with any Application Support server, an OTMA server's CDS directory and its DCE principal name are supplied through SYSIN. (See Figure 20 on page 91.)

Environment variables supply configuration information for OTMA's use of XCF. Make sure that the environment variables listed in the example worksheet (see "AS IMS with OTMA Worksheet" on page 80) are set in the `envar` file or that the default value, if available, is the value you want. If you are using transactional RPCs, the environment variable `_ASU_RESOURCE_MGR_NAME` identifies your Application Support server to RRS.

Table 6 on page 26 lists these environment variables and shows the format for setting them in the `envar` file. See "Step 4: Create the Environment Variable File" on page 23 for information on building your `envar` file.

XCF requires that Application Support and IMS use the same group name and that Application Support must know the IMS member name. In addition, you must specify the environment variable `_ASU_XCF_SERVER_MEMBER_NAME` to identify your Application Support server to XCF.

For more information about OTMA and XCF, including configuration, see the following:

- *IMS/ESA Open Transaction Manager Access Guide*
- *z/OS MVS Programming: Sysplex Services Guide*
- *z/OS MVS Programming: Sysplex Services Reference*

- *z/OS MVS Programming: Resource Recovery*

Step 3: Configure Encina for Use with Application Support

If you are using the OTMA adaptor for transactional RPCs, you need to configure Encina for use with Application Support. The Encina components require the use of a CDS directory structure. You are recommended to use the default Encina root directory name `./encina`. You also need a directory entry called `trpc` in that directory. For example, if you use the default root directory, this directory entry is called `./encina/trpc`. If these directory entries do not exist in a cell, you need to create them. See the z/OS Encina Toolkit Executive for information about creating `./encina` and `./encina/trpc`.

If you want to change the default root directory name, you need to set the environment variable `ENCINA_CDS_ROOT` to the name you want to use. For more information about using an Encina directory other than the default, see the Encina Toolkit administrator's documentation for your system.

The AS server principal needs write access to the `./encina` directory and write and insert access to the `./encina/trpc` directory. You are recommended to create a DCE group, for example, `encina-servers`, add Application Support to that group, and give that group write authority to the directories. General read access must be given because principals of other servers may need to read entries in these directories.

Step 4: Set IMS OTMA Security

If you wish to use RACF to secure access of IMS transactions and commands you should use:

`/SECURE OTMA,PROFILE`

Operational Considerations When Using AS IMS with OTMA

- The PROC `ASUVSO` is the startup procedure for the AS IMS OTMA server. "Startup Procedure for DCE AS IMS with OTMA" on page 90 documents this startup procedure.
- The AS IMS OTMA server module and stub library must run APF-authorized.
- You can use all existing AS IMS operator commands and administration program panels with the OTMA adaptor. The `ASUADMIN` program ISPF panels allow the definition of an OTMA-based server and display unique OTMA status information. (See Chapter 17, "Using the `ASUADMIN` Administration Program" on page 97.)
- You can use Encina `tkadmin` commands to determine the state of the server and Encina transactions. See Chapter 20, "Using the `tkadmin` Command with the AS Server" on page 135 and the Administrator's Guide for the Encina Toolkit for your system for more information.
- Environment variables let you specify OTMA-unique environment definitions, such as XCF member and group names. See Table 6 on page 26 for more information.
- If you are using transactional RPCs, RRS must be started before the AS IMS OTMA server is started.

Chapter 7. Configuring the AS IMS APPC Server

This chapter describes the necessary IMS resource definitions to use the APPC adaptor for communications between the Application Support server and IMS. For information about deciding which adaptor to use, see Table 1 on page 9.

Defining IMS resources is done through special macro instructions. In the following sections that describe these definitions, sample macro statements defining resource attributes are listed. If the definition of an attribute is not described, it is either not relevant to DCE AS IMS, or it may assume its default value.

Definitions for DCE AS IMS with APPC

An APPC/MVS transaction program (TP) requires definitions to MVS, VTAM, and RACF. This chapter takes you through the steps you need to perform to configure IMS for the Application Support server when you are using the APPC communications method:

- Step 1. Specify APPCPMxx Member in SYS1.PARMLIB.
- Step 2. Define AS IMS to VTAM.
- Step 3. Define AS IMS to RACF.
- Step 4. Check settings of AS IMS with APPC environment variables.
- Step 5. Configure and install the IMS DFSLUEE0 exit routine if you want Application Support to be able to accept and understand LTERM and MOD Name data.

The following sections describe these steps.

Note: This section does not show the APPC definitions necessary for IMS. See the chapter about administering APPC/IMS and LU 6.2 devices in the *IMS/ESA Administration Guide: Transaction Manager*.

Step 1: Specify APPCPMxx Member in SYS1.PARMLIB

APPCPMxx members define local logical units to APPC/MVS. They also set up a correspondence between local LUs and transaction schedulers, and, optionally, define LUs that are not to be associated with schedulers ("NOSCHED" LUs, such as AS IMS). Specify an LUADD statement for each logical unit defined in the member as follows:

- The ACBNAME parameter defines the name of the logical unit. In the example shown in Figure 8 on page 44, MVSIMS02 is the logical unit name associated with AS IMS. The LU name specified with ACBNAME must be the same name specified on the `_ASU_SERVER_LUNAME` environment variable. It is the same name specified in step 5.

Note: AS IMS must be the only program in the system using this name and protected with RACF. See "Step 5: Configure the IMS DFSLUEE0 Exit Routine" on page 46 for more information on configuring DFSLUEE0. See Table 6 on page 26 for information on `_ASU_SERVER_LUNAME`.

- The NOSCHED, TPLEVEL, and TPDATA parameters must be specified for AS IMS. The TPLEVEL parameter must have a value of SYSTEM. The TPDATA parameter specifies the name of a VSAM KSDS data set (SYS1.APPCTP in the example in Figure 8 on page 44) that contains TP profiles along with a database token used for verifying access authority to TP profiles.
- Each member may also contain a SIDEINFO statement that specifies the name of the side information file used for outbound allocate requests. The side information file is a VSAM KSDS data set (shown as SYS1.APPCSI in Figure 8 on page 44) allocated through the IDCAMS utility program. The MVS ATBSEDFMU utility program is then used to add the side information to this file, using the SIADD utility statement, as shown in Figure 9 on page 44.

Figure 8 on page 44 shows an example LUADD statement. The MVS ATBSDLF MU utility program is used to add the database token using the DBMODIFY utility statement. For a NOSCHED LU (such as AS IMS), APPC/MVS uses only the data set's database token and not any profile information.

```
LUADD ACBNAME(MVSIMS02) NOSCHED TPLEVEL(SYSTEM) TPDATA(SYS1.APPCTP)
SIDEINFO DATASET(SYS1.APPCSI)
```

Figure 8. LUADD statement

If you specify a side information file with the SIDEINFO statement, you must also specify a SIADD statement as follows:

- DESTNAME specifies the symbolic destination name for the partner TP (in this case, IMS). This is the key by which APPC/MVS finds the side information data in the side information file. AS IMS uses a name that is supplied with the `_ASU_SYMBOLIC_NAME` environment variable (see Table 6 on page 26). In the example shown in Figure 9, the symbolic destination name is IMS1.
- TPNAME specifies the name of the TP. This name can be anything, as AS IMS must override this value based on the TP name specified in each incoming RPC. In the example the TP name is IMSTP1.
- PARTNER_LU specifies the LU name defined by the installation for IMS. In the example, the LU name is IMSMVS02.

```
SIADD
  DESTNAME(IMS1)
  TPNAME(IMSTP1)
  PARTNER_LU(IMSMVS02)
```

Figure 9. Example SIADD statement

When the APPC address space is started, the operator tailors its execution by specifying the suffixes of one or more APPCPMxx members, thus defining to APPC the LU names of which it needs to be aware.

See *z/OS MVS Planning: APPC/MVS Management* for detailed information on the APPCPMxx PARMLIB member and creating the side information file.

Step 2: Define AS IMS to VTAM

Both LUs must be defined to VTAM.

The AS IMS LU must be defined to VTAM with a VTAM application (APPL) definition statement, shown in Figure 10 on page 45. The APPL statement resides in a SYS1.VTAMLST member. The APPL statement is defined as follows:

- The value preceding the APPL keyword and the ACBNAME value (both shown as MVSIMS02 in the example shown in Figure 10 on page 45) must be the same as the value specified on the ACBNAME keyword in the APPCPMxx PARMLIB member that defines the AS IMS LU to APPC/MVS. See “Step 1: Specify APPCPMxx Member in SYS1.PARMLIB” on page 43.
- The APPL statement supports many more keywords. Accept the default values assigned to these keywords.

```

ASIMS    VBUILD TYPE=APPL
MVSIMS02 APPL ACBNAME=MVSIMS02,           X
          APPC=YES,                       X
          SECACPT=ALREADYV

```

Figure 10. Example APPL definition statement

For detailed information on coding the APPL statement and including it in SYS1.VTAMLST, see:

- *z/OS MVS Planning: APPC/MVS Management*
- *z/OS Communications Server: SNA Network Implementation Guide*
- *z/OS Communications Server: SNA Resource Definition Reference*
- *Planning for NetView, NCP, and VTAM*

Step 3: Define AS IMS to RACF

An APPC/MVS application contains a number of different resources: logical units, transaction programs, sessions, conversations, and data sets. To protect each type of resource, RACF supplies a resource class. By defining a RACF profile for the resource in the corresponding class, the application administrator can control access to the resource. Therefore, the administrator can tailor the level of security necessary to protect the resources of the application. This tailoring is transparent to AS IMS. That is, AS IMS is a standard APPC/MVS application program. It does not interface directly with RACF, but RACF controls that affect APPC/MVS applications can determine AS IMS access to IMS and individual IMS transactions.

The following RACF security classes apply to APPC/MVS:

```

APPCLU    authorizes session between LUs
APPCPORT authorizes user access from specific LU
APPCSERV authorizes user access to specific servers
APPCSI   authorizes user access to APPC/MVS side information
APPCTP   authorizes user access to APPC/MVS TP profile
APPL     authorizes user access to specific local LU
FACILITY authorizes user access to database tokens
PROGRAM  authorizes ATBSDFMU access to data set
VTAMAPPL authorizes a program to open a VTAM ACB for an LU

```

For Application Support, you need to use the APPCLU class to define LU-to-LU security for both the Application Support LU and the IMS/APPC LU using a matching LU session key. You create RACF profiles in the APPCLU class as follows:

```
RDEFINE APPCLU network-id.local-lu-name.partner-lu-name SESSION(SESSKEY(sessionkey) CONVSEC(ALREADY)) UACC(access)
```

For example, where the network ID is AAA, you could use:

```
RDEFINE APPCLU AAA.MVSIMS02.IMSMVS02 SESSION(SESSKEY(1234CD5)CONVSEC(ALREADYV))
```

Define the partner as follows:

```
RDEFINE APPCLU AAA.IMSMVS02.MVSIMS02 SESSION(SESSKEY(1234CD5)CONVSEC(ALREADYV))
```

Note that the session key is a password whose expiration interval depends on the value of the global RACF SESSIONINTERVAL setting.

When you are ready to start using the protection defined in the APPCLU profiles for each LU, the security administrator should activate the APPCLU class:

```
SETROPTS CLASSACT (APPCLU)
```

Any time an APPCLU profile changes, the following command refreshes the profile so that changes take effect:

```
F NET,PROFILES, ID=local-lu-name
```

For detailed information on APPC/MVS application security, see the *z/OS MVS Planning: APPC/MVS Management*.

For general information on RACF, see:

- *z/OS SecureWay Security Server RACF Security Administrator's Guide*
- *z/OS SecureWay Security Server RACF Command Language Reference*

Step 4: Ensure Environment Variables for AS IMS with APPC Have Been Set

Make sure that the environment variables listed in the example worksheet (see “AS IMS with APPC Worksheet” on page 79) are set in the `envar` file, or that the default value, if available, is the value you want.

Table 6 on page 26 lists these environment variables and shows the format for setting them in the `envar` file. See “Step 4: Create the Environment Variable File” on page 23 for information on building your `envar` file. For more information on specifying environment variables for a z/OS program, see the *z/OS DCE Administration Guide*.

Step 5: Configure the IMS DFSLUEE0 Exit Routine

Note: This step is optional. Do this step only if you want Application Support to be able to accept and understand MOD Name and LTERM data. If you do not need this, continue with Chapter 9, “Creating the DCE Account for the AS Server” on page 51. The DFSLUEE0 exit routine is called by IMS prior to inserting an inbound message segment into the message queue or removing an outbound message segment from the message queue.

The DFSLUEE0 exit is supplied as System/370™ assembler source as member ASULUEE0 in data set *ASPREFX.SASUSAMP*. The ASULUEE0 assembler source provided with the Application Support product must be assembled and link-edited into the IMS.RESLIB (or equivalent STEPLIB) data set as module DFSLUEE0 by the installation system programmer. The IMS exit routine structure does not allow the installation of multiple exit routines at the same exit point. If you have added code to the existing DFSLUEE0 exit routine, you must customize your code to add the logic supplied in the ASULUEE0 source file. If you have not added code to the DFSLUEE0 exit routine, you may use the ASULUEE0 source file as a complete replacement for DFSLUEE0 after making the changes described in the following. You must examine the source and make appropriate changes based on the IMS level you are using.

The ASULUEE0 exit routine contains a table of LU names whose entries must match the local LU names of all the instances of the Application Support server that are communicating with the host IMS system through APPC. The LU name table is at label ASUJLTAB. The table is used by the exit routine logic to identify conversations originated by or directed to the AS IMS LU. This is necessary so that the AS IMS exit code does not try to change APPC/IMS conversations that do not involve AS IMS. The value specified at the ASUJLTAB label in ASULUEE0 must match the value specified on the `_ASU_SERVER_LUNAME` environment variable in the `envar` file. See Appendix A, “ASULUEE0 Exit Routine” on page 159 for a listing of the exit code.

Chapter 8. Configuring the AS IMS ISC Server

This chapter describes the necessary IMS resource definitions to use the ISC adaptor for communications between the Application Support server and IMS. For information about deciding which adaptor to use, see Table 1 on page 9.

Defining IMS resources is done through special macro instructions. In the following sections that describe these definitions, sample macro statements defining resource attributes are listed. If the definition of an attribute is not described, it is either not relevant to DCE AS IMS, or it may assume its default value.

Definitions for AS IMS with ISC

To define AS IMS with ISC to the Application Support server, there are a number of steps you must follow:

- Step 1. Specify IMS Definitions for TYPE, TERMINAL and VTAMPOOL
- Step 2. Specify VTAM definitions in the VTAM definition library
- Step 3. Define DCE AS IMS with ISC attributes to the CDS
- Step 4. Get access to IMS programs

This chapter describes these steps. See *IMS/ESA Administration Guide: Transaction Manager* for more information on statically defining an ISC node to IMS.

Step 1: Specify IMS Definitions for TYPE, TERMINAL, and VTAMPOOL

You must define the ISC connection between IMS and the Application Support server during IMS Stage 1 system definition. The connection is defined as LUTYPE6 (ISC) with parallel sessions to IMS. The Application Support server must be defined to IMS as a SINGLE1 ISC node. Any other definition results in data streams that the Application Support server does not support. Figure 11 is an example of an IMS TYPE and TERMINAL definition:

```
TYPE      UNITYPE=LUTYPE6
TERMINAL  NAME=ASIMS,
          COMPT1=(SINGLE1,VLVB),
          OPTIONS=(TRANRESP,MTOMSG),
          SESSION=5
```

Figure 11. Example of an IMS TYPE and TERMINAL Definition

The Application Support server maps inbound RPC calls to one of the available parallel sessions. The number of available parallel sessions is determined by the length of the prefix you specify. (The SESSION entry in the IMS TERMINAL definition, SESSION=5 in the preceding example, identifies the number of active sessions you can have at one time.) IMS requires that each of the source end points be identified as a logical device by a unique name. Note that the names used to identify the end points do not represent actual SNA LU names, but rather are treated as a pool of distinct session end points managed by the Application Support server. To that extent, the names defined to IMS must match the names constructed by the Application Support server (based on its configuration information).

To minimize the configuration task, the Application Support server assumes that these names (session endpoints) consist of a constant prefix and a suffix beginning with 0 for the first session endpoint and incremented by 1 for each subsequent endpoint up to the maximum number of sessions. Hexadecimal

values are used in the suffix. By convention, names must be 8 characters long: The prefix may be 4 to 7 characters long. The suffix is padded with zeroes on the left as required to total 8 characters when combined with the prefix. The name prefix is specified within the Application Support server directory entry.

Note: The sessions must be defined in consecutive hexadecimal order. For example, if the suffixes of the session numbers are two digits long, they must be numbered in the following sequence:

00, 01, 02, ..., 09, 0A, 0B, ..., 0F, 10, 11, 12, ..., FF

The algorithm for establishing session numbers can tolerate no more than 10-session gaps in the session numbering.

The Application Support server simply continues to bind sessions until VTAM or IMS indicates that additional sessions cannot be bound, or until another session endpoint name cannot be generated (all suffix values have been used).

In the IMS system definition, the full list of potential logical devices must be explicitly defined, similar to the following example in Figure 12.

```
VTAMPOOL
SUBPOOL  NAME=IBMAPPS0
NAME     IBMAPPS0
SUBPOOL  NAME=IBMAPPS1
NAME     IBMAPPS1
SUBPOOL  NAME=IBMAPPS2
NAME     IBMAPPS2
SUBPOOL  NAME=IBMAPPS3
NAME     IBMAPPS3
SUBPOOL  NAME=IBMAPPS4
NAME     IBMAPPS4
SUBPOOL  NAME=IBMAPPS5
NAME     IBMAPPS5
SUBPOOL  NAME=IBMAPPS6
NAME     IBMAPPS6
SUBPOOL  NAME=IBMAPPS7
NAME     IBMAPPS7
SUBPOOL  NAME=IBMAPPS8
NAME     IBMAPPS8
SUBPOOL  NAME=IBMAPPS9
NAME     IBMAPPS9
SUBPOOL  NAME=IBMAPPSA
NAME     IBMAPPSA
SUBPOOL  NAME=IBMAPPSB
NAME     IBMAPPSB
SUBPOOL  NAME=IBMAPPSC
NAME     IBMAPPSC
SUBPOOL  NAME=IBMAPPD
NAME     IBMAPPD
SUBPOOL  NAME=IBMAPPSE
NAME     IBMAPPSE
SUBPOOL  NAME=IBMAPPF
NAME     IBMAPPF
```

Figure 12. Example of Defining Logical Devices

In this example, the prefix **IBMAPPS** was specified in the CDS entry for this server, and the **SESSION** parameter of the preceding **TERMINAL** macro had the value of **5**.

Note: Use the **/STOP SUBPOOL IMS** command to disconnect IMS from the Application Support server instead of the **/STOP LTERM IMS** command. Using **/STOP LTERM** results in an error condition.

Step 2: Specify VTAM Definitions in the VTAM Definition Library

For attachment to IMS, you must define both IMS and the Application Support server to VTAM. These definitions are part of the VTAM definition library (SYS1.VTAMLST).

Figure 13 is an example of APPL definitions in VTAM:

```
IOEIMS4 APPL ACBNAME=IOEIMS4,AUTH=(ACQ), X
            PARSESS=YES, X
            EAS=1200
ASIMS APPL ACBNAME=ASIMS, X
            PARSESS=YES, X
            EAS=100, X
            DLOGMOD=LU6NEGPS, X
            MODETAB=MTAPPLS, X
            PRTCT=OUR1PASS
```

Figure 13. Example of APPL Definitions in VTAM

The first **ACBNAME** value (shown as **IOEIMS4** in the preceding example) must be the same as the **APPLID** name on the **COMM** macro in the IMS Stage 1 definition. The value of **PRTCT** must be the same as the value of the VTAM password in the data set that is pointed to by the startup procedure for DCE AS IMS, as described in “Startup Procedure for DCE AS IMS with ISC” on page 92. In addition to the APPL definitions, the VTAM mode table must contain an entry for IMS ISC sessions.

Following is an example of an entry in a VTAM mode table:

Note: Except for **LOGMODE** and **RUSIZES**, the other definitions must be coded exactly as illustrated in Figure 14

```
LU6NEGPS MODEENT LOGMODE=LU6NEGPS,
            TYPE=0, NEGOTIATED BIND X
            FM PROF=X'12', FM PROFILE 18 X
            TS PROF=X'04', TS PROFILE 4 X
            PRIPROT=X'B1', PRIM NAU PROTO (EX/DEF, EB) X
            SEC PROT=X'B1', SEC NAU PROTO (EX/DEF, EB) X
            COM PROT=X'70A0', COMMON NAU PROTO (BETB,COLDSTART) X
            RUSIZES=X'F8F8', SEND/ RCVE 3840 BYTES X
            PSERVIC=X'060038000000380000000000'
```

Figure 14. Example of an Entry in a VTAM Mode Table

Step 3: Define DCE AS IMS with ISC Attributes to the CDS

The IMS and VTAM definition data described in the previous sections are also used when you **define** the DCE AS IMS attributes to the Cell Directory Service (CDS). This is described in Chapter 13, “Defining the Application Support Server to CDS” on page 73. The value of the PRTCT definition is used again when you start the Application Support server.

Keep the following information for later use:

- VTAM ACB name of the Application Support server
In the Create an IMS Application Support Server Definition panel, this must match the value specified in the **Application Support Server LU Name** field.
- VTAM ACB name of IMS
In the Create an IMS Application Support Server Definition panel, this must match the value specified in the **IMS LU Name** field.
- Subpool identifier prefix
In the Create an IMS Application Support Server Definition panel, this must match the value specified in the **Application Support Server Device Prefix** field.
- Label of the MODEENT macro (Mode name)
In the Create an IMS Application Support Server Definition panel, this must match the value specified in the **Session Logon Mode Name** field
- The PRTCT definition.
You use the value specified in this definition in the SYSIN of the JCL that starts the Application Support server.

Step 4: Get Access to IMS Programs

The IMS server application programs must be identified to IMS by matching definitions in the IMS Gen (TRANSACT and APPLCTN macros) and by the proper PSB/ACB definitions.

You should concern yourself with securing access to the IMS transactions. You have two choices in addressing transaction security in the Application Support server environment:

- You can restrict a set of transactions to the VTAM subpool associated with the server and thus block users that submit their messages through LTERMs other than the specified subpool. You must ensure, through the DCE services, that only authorized users can access this server. However, to provide better security control, consider the next method.
- In IMS Version 4 and later, you can also use the sign-on security features of IMS, where a RACF (or equivalent) user ID is granted access to a set of transactions. In this case, the DCE client application must always issue a /SIGN ON call immediately following the OPEN SESSION call, thus identifying the user with RACF ID and password. IMS then uses this user ID to grant or deny access to IMS transactions.

Whichever method you use, see the *IMS/ESA Administration Guide: System* for details of the IMS aspects of this definition.

The transaction codes are specified to DCE through the IDL file. It is recommended that only an authorized user be in charge of compiling all IDL definitions for the interfaces and installing the load modules in the stub library.

Chapter 9. Creating the DCE Account for the AS Server

In DCE, noninteractive participants such as the Application Support server must authenticate themselves (that is, log in to DCE) to be able to run under their own identities. To run under its own DCE identity, a DCE account has to be created for the Application Support server. This chapter takes you through the steps you need to perform to create the DCE account for the Application Support server:

- Step 1. Update the keytab file for the Application Support server.
- Step 2. Give the Application Support server access to the DCE endpoint map.
- Step 3. Select the group for the Application Support server.
- Step 4. Create a DCE account.
- Step 5. Ensure certificate lifetime across cells.

Before You Begin Creating a DCE Account: Before you can create a DCE account for the Application Support server, the following must be done:

- Your DCE administrator must have already configured the cell in which to create the account.
- The Security group and organization of which the Application Support server will be a member must have been created.

“Step 2: Give the AS Server Access to the DCE Endpoint Map” on page 52 discusses the alternatives in selecting the Security group to which the Application Support server belongs.

Step 1: Update the Keytab File for the Application Support Server

A server that runs under its own identity (for example, the Application Support server) stores its password in a local key table file, called the **keytab** file. The keytab file facilitates the entry of passwords when the server logs in to DCE.

Step 1a: Set the `_ASU_KEYTAB_FILE` Environment Variable

The `_ASU_KEYTAB_FILE` environment variable sets the HFS pathname of the keytab file of the Application Support server. You can set this variable in the Application Support server's `envar` file or through any other means the *z/OS DCE Administration Guide* describes. For example, if the pathname of the keytab file is `/var/AS/server_1/ktabfile`, make the following entry in the `envar` file:

```
_ASU_KEYTAB_FILE=/var/AS/server_1/ktabfile
```

If this variable is not set, the Application Support server uses the default keytab file (`/krb5/v5srvtab`) when it is started.

Step 1b: Secure the Keytab File

Because the keytab file contains the password of the Application Support server principal, it must be protected from access by unauthorized users. The keytab file is an HFS file and can be protected by manipulating the permission bits of the file. See the *z/OS UNIX System Services User's Guide* for more information on manipulating permission bits.

Only the following should have read and write access to the keytab file:

- The superuser
- The MVS user ID under which the AS server runs
- The Application Support server administrator

Step 2: Give the AS Server Access to the DCE Endpoint Map

The Application Support server registers or unregisters the management interfaces that it supports with the DCE endpoint map. The Application Support server must have the appropriate permissions to the DCE endpoint map to be able to perform these tasks. There are several ways to do this:

1. You can define the Application Support server's access to the DCE endpoint map by editing the ACL associated with the DCE endpoint map object and adding an ACL entry for the Application Support server, or the group to which the Application Support server belongs. The permission that Application Support servers require to access the DCE endpoint map is the **s** (server) permission, which allows the server principal to register and unregister endpoints in the DCE endpoint map.
2. In z/OS DCE, the **subsys/dce/rpc-server-group** Security group is created during the DCE configuration of the z/OS host system. This group already has the required permission to allow modification of the DCE endpoint map. With the **subsys/dce/rpc-server-group**, you only need to make each Application Support server a member of this group to gain access to the DCE endpoint map. This is discussed in "Step 4: Create a DCE Account for the Application Support Server."

Step 3: Select the Group for the Application Support Server

For administrative purposes, you may want to create Security groups composed of Application Support servers that share a common administrative purpose, for example, Application Support servers that have a common set of users. When creating or selecting this group, keep in mind that you can use the Security group not only to facilitate the administration of these Application Support servers, but also to facilitate access to the DCE endpoint map.

If you choose to create different administrative groups for the Application Support servers, remember that the Application Support server can be a member of multiple groups. The group that is specified when you create the DCE account is referred to as the **primary** group. The Application Support server can be made a member of other groups besides the primary group by using the DCE administrative interface (**dcecp**) group **add** command.

Thus, by using Security groups, access to the DCE endpoint map can be achieved in either of two ways:

- Use an administrative group as the primary group for the Application Support server and then make the Application Support server also a member of the **subsys/dce/rpc-server-group** using the DCE administrative interface (**dcecp**).
- Use an administrative group as the primary group for the Application Support server and give that group the appropriate permission to the DCE endpoint map. Giving this permission is similar to that described in "Step 2: Give the AS Server Access to the DCE Endpoint Map." Creating and adding members to groups is performed by the cell administrator.

Step 4: Create a DCE Account for the Application Support Server

This section describes how to use the DCE administrative interface (**dcecp**) to create the DCE account for the Application Support server, and add its password to the keytab file.

The procedure for creating the DCE account for the Application Support server is summarized and illustrated in the following steps:

Note: Make sure you are logged in to DCE as the DCE administrator before performing these steps. Because the DCE administrative interface (**dcecp**) is itself a distributed application, all of the following

steps except the `rgy_edit ktadd` can be performed on any host system in the cell that has the DCE administrative interface.

Creating the DCE principal ID of the Application Support server:

1. Start the DCE administrative interface from TSO or from MVS as follows:

```
dcecp
```

Note: From this point, steps include the prompt `dcecp>` in example lines.

2. Use the **principal** subcommand to create the Application Support server principal. For example, if the principal name of the Application Support server is `p_server_1`:

```
dcecp> principal create p_server_1
```

This assigns a default UNIX number, full name, and object creation quota. If you want to enter values other than the defaults, see the Security part of the *z/OS DCE Administration Guide* for details.

Adding the Application Support server to a Security group:

3. Decide the DCE Security group to which the Application Support server will be added as a member. This is discussed in “Step 3: Select the Group for the Application Support Server” on page 52.

In the examples that follow, the Application Support server is added to a group, `asgrp` as its **primary** group. Then, the Application Support server is added as a member of `subsys/dce/rpc-server-group`, to give it access to the DCE endpoint map.

In these examples, it is assumed that the group `asgrp` and the organization `ibm` have not been previously created in the registry.

Creating the group asgrp:

4. Add the group `asgrp`:

```
dcecp> group create asgrp
```

This assigns a default UNIX number and full name and includes the group `asgrp` in the project list. If you want to enter values other than the defaults, see the Security part of *z/OS DCE Administration Guide* for details.

5. Add the principal to the group. For example, to add the principal `p_server_1` to `asgrp`, enter:

```
dcecp> group add asgrp -member p_server_1
```

Creating the organization ibm:

6. Add the organization to which the user will belong:

```
dcecp> organization create ibm
```

This assigns a default UNIX number and full name. If you want to enter values other than the defaults, see the Security part of *z/OS DCE Administration Guide* for details.

7. Add the principal to the organization. For example, to add the principal `p_server_1` to the organization `ibm`, enter:

```
dcecp> organization add ibm -member p_server_1
```

Creating the DCE account:

8. Add the account for the Application Support server. For example, for the Application Support server whose principal name is `p_server_1`:

```
dcecp> account create p_server_1 -group asgrp -organization ibm \  
-password pword -mypwd -dce-
```

You use the *pwd* in the preceding example again when you specify the password in the keytab file. This is described in Step 10 on page 54 of this procedure. The *-dce-* in the example is the DCE administrator's password.

Adding the Application Support server to subsys/dce/rpc-server-group:

9. Add the Application Support server as a member of the RPC server group.

```
dcecp> group add subsys/dce/rpc-server-group -member p_server_1
```

Adding the Application Support server's password to the keytab file:

10. Add the Application Support server's password (also called the key) to the keytab file using the **ktadd** subcommand of the Registry Editor.

Note: This step must be performed on the host system where the Application Support server is installed because the **ktadd** command creates and uses the keytab file on the host system where it is run.

To start the Registry Editor from TSO, enter:

```
rgyedit
```

From MVS, enter:

```
rgy_edit
```

Add the password.

Note: The following example includes the `rgy_edit=>` prompt.

For example, if the password of the Application Support server *p_server_1* is *pwd*, enter:

```
rgy_edit=> ktadd -p p_server_1 -pw pwd -f /var/AS/server_1/keytabfile
```

where */var/AS/server_1/keytabfile* is the path name of the keytab file and *pwd* is the password that you specified when you created the account of the Application Support server in Step 8 on page 53.

You must create one entry for each instance of the Application Support server in a keytab file. Each Application Support server can also use a different keytab file, in which case, you need to create separate keytab files for each instance of the Application Support server. However, the recommended method is a single keytab file for all the instances of the Application Support server.

You can elect not to create any keytab file for the Application Support server, in which case, the default keytab file (**/krb5/v5srvtab**) is used. However, it is recommended that you create a different keytab file for the Application Support servers because only very few privileged servers are allowed to add keys to the default keytab file.

It is recommended that the keytab file reside under the Application Support server's directory. You may want to make the Application Support server's MVS user ID the owner of the keytab file. You must ensure that the Application Support server has read/write access to the file.

Step 5: Ensure Certificate Lifetimes Across Cells

If the Application Support server is going to serve DCE clients in foreign cells, ensure that the minimum ticket lifetime property of the cell where the Application Support server is running is less than the default certificate lifetime of the foreign cells. For more information on certificate lifetimes, see the *z/OS DCE Administration Guide*.

Chapter 10. Creating Administrator and User Accounts

The DCE Security Service must authenticate and authorize all Application Support server users, including the administrator, to enable them to access the Application Support server. Application Support server users are DCE clients who access the Application Support server. A DCE account must be created for each Application Support server user.

The Application Support server user also accesses non-DCE resources such as the CICS or IMS regions and their associated databases. Because of this, a relationship (referred to as a mapping) must be established between the user's DCE principal name and the user's MVS user ID. To accomplish these tasks, perform the following steps:

- Step 1. Register users with the MVS security system
- Step 2. Create a DCE account for each Application Support server user
- Step 3. Map the DCE principal names to the local MVS user IDs
- Step 4. Map foreign users to MVS user IDs

This chapter describes these steps. It also contains information on changing the Identity Mapping file in “Modifying and Deleting Identity Mapping Entries” on page 60.

Step 1: Register Users with the MVS Security Subsystem

Clients of the Application Support server ultimately access CICS or IMS regions and other non-DCE resources these subsystems own. The local MVS security subsystem (for example, RACF) controls access to these resources. Thus, all users of the Application Support server must be registered with the specific security subsystem on the MVS host. For information on registering users on the local security subsystem, see the documentation the Security software on your host provides.

Step 2: Create DCE Accounts for Application Support Server Users and Administrators

Note: DCE User accounts must be created by a user with DCE administrator privileges.

The procedures for creating a DCE account described here assume that the principal, group, and organization have not been created prior to the creation of the account. Skip the relevant steps if the principal, group, and organization have previously been created for the user.

To create a DCE account, follow these steps:

1. Start the DCE administrative interface:
dcecp
2. Use the **principal** subcommand to create the principal for the user.

Note: The examples that follow include the `dcecp>` prompt.

For example, if the Application Support server user is named *smith*:

```
dcecp> principal create smith
```

This assigns a default UNIX number, full name, and object creation quota. If you want to enter values other than the defaults, see the Security part of the *z/OS DCE Administration Guide* for details.

3. You have to perform this step only if the group or groups have not been previously added to the registry. Add the group (or groups) to which the user will belong. For example, if the name of the group is *usergrp*:

```
dcecp> group create usergrp
```

This assigns a default UNIX number and full name and includes the group *usergrp* in the project list. If you want to enter values other than the defaults, see the Security part of the *z/OS DCE Administration Guide* for details.

4. Add the principal to the group. For example, to add the principal *smith* to *usergrp*:

```
dcecp> group add usergrp -member smith
```

5. You have to perform this step only if the organization has not previously been created in the registry. Add the organization to which the user will belong. For example, if the organization is named *ibm*:

```
dcecp> organization create ibm
```

This assigns a default UNIX number and full name. If you want to enter values other than the defaults, see the Security part of the *z/OS DCE Administration Guide* for details.

6. Add the principal to the organization. For example, to add the principal *smith* to the organization, *ibm*:

```
dcecp> organization add ibm -member smith
```

7. Add the account for the user. For example, for the user whose principal name is *smith*:

```
dcecp> account create smith -group usergrp -organization ibm \  
-password pword -mypwd -dce-
```

In the preceding example, you must supply *pword*, a user's password associated with the newly created account and *-dce-*, the administrator's password.

After creating the DCE accounts for the users, their DCE principal names must be mapped to their z/OS user IDs. This activity is described in “Step 3: Mapping DCE Principal Names to MVS User IDs.”

Step 3: Mapping DCE Principal Names to MVS User IDs

Note: This must be done by someone with Application Support server administrator privileges.

The Application Support server enables DCE clients to access non-DCE resources such as CICS and IMS regions and the associated databases. The CICS or IMS subsystem treats calls from DCE clients the same way it treats calls from CICS or IMS clients.

The local security subsystem determines whether the client is authorized to use the resource. Because the local security subsystem is not integrated with the DCE Security service, the local security subsystem cannot recognize DCE principal names. To overcome this, the administrator has to establish a mapping between a user's DCE principal name and MVS user ID.

There are two methods of mapping DCE principal names to MVS user IDs:

- The Identity Mapping file
- RACF

If you are already using RACF to provide security for your organization, you may want to use RACF for identity mapping. If you are not already using RACF, you probably want to use the Identity Mapping file. “Mapping DCE Principal Names to MVS User IDs with the Identity Mapping File” on page 57 describes the Identity Mapping file method. “Mapping DCE Principal Names to MVS User IDs with RACF” on page 60 describes the RACF method.

Mapping DCE Principal Names to MVS User IDs with the Identity Mapping File

To map DCE principal names to MVS user IDs, perform the following steps:

- Step 3A. Create the **Identity Mapping File**.
- Step 3B. Set the **_ASU_IDMAP** environment variable.
- Step 3C. Map foreign users to MVS user IDs.

These procedures are described in the following sections.

Step 3A: Create the Application Support Identity Mapping File: The Identity Mapping file is a text file that the administrator creates and maintains. This must be created as an HFS file.

Note: The Application Support server administrator must also map his DCE principal name to his MVS user ID.

The Identity Mapping file contains one or more identity mapping declarations and has the following general format:

```
DCE-principal-name1  
MVS-user-ID1  
Application-Support-server-name
```

```
DCE-principal-name2  
MVS-user-ID2
```

...

This format illustrates the two types of entries that can exist on the Identity Mapping file. The first entry has three elements: DCE principal name, MVS user ID, and Application Support server name. The second entry only has two elements: DCE principal name and MVS user ID. These are explained later in this chapter. A blank line is required between entries. The following list explains each element in an identity mapping entry:

<i>DCE-principal-name</i>	Is the client's DCE identity. This may either be a simple DCE principal name (for clients within the cell) or a fully qualified global name (for clients within and outside the cell). The DCE principal name can be up to 256 characters in length.
<i>MVS-user-ID</i>	Is the MVS user ID of the client. All potential Application Support server clients must have user IDs on the MVS host where the Application Support server is running.
<i>Application-Support-server-name</i>	Is the name of the CDS object that represents a specific Application Support server. This is optional. If the mapping entry includes the name of an Application Support server, the mapping is valid only for that instance of the Application Support server. If the entry does not include the name of an Application Support server, the mapping is valid for all Application Support servers that are running on the host system.

The first entry type (with three elements) is known as the **specific mapping** to an Application Support server. The second entry type (with two elements) is known as the **default mapping** for an Application Support server user. You can have one or both of these entry types in your Identity Mapping file.

Although not required, it is recommended that all Application Support servers on a host system share the same Identity Mapping file. Figure 15 on page 58 shows a sample Identity Mapping file.

```
smi th
TSSMITH
././AS/server_1

jones
TSJONES
```

Figure 15. Example Identity Mapping File

In the first entry, the DCE principal name **smith** is mapped to the MVS user ID **TSSMITH**. This mapping is effective only when accessing the Application Support server whose CDS name is **././AS/server_1**. In the second entry, the DCE principal name **jones** is mapped to the MVS user ID **TSJONES**. This mapping is effective when accessing **any** Application Support server on the host system.

If a user has both specific mapping and default mapping entries with the same principal in the file, the specific mapping entry overrides the default mapping entry. Consider an Identity Mapping file that has the following entries:

```
smi th
TSSMITH

smi th
TSJOHN
././AS/server_1
```

The second entry (specific) overrides the first entry when it is run from **server_1**.

Each user can only have one default mapping for every DCE principal in the file. For example, the following mapping entries are **not** allowed:

```
smi th
TSSMITH

smi th
TSJOHN
```

Also, each user can only have one specific mapping entry to a particular Application Support server for every DCE principal in this file. For example, the following entries are **not** allowed:

```
smi th
TSSMITH
././AS/server_1

smi th
TSJOHN
././AS/server_1
```

Identity Mapping File Processing: When the Application Support server is started, the Identity Mapping file is read and stored in the Application Support server storage area. If severe errors are encountered, for example, if the Identity Mapping file cannot be read, the initialization of the Application Support server fails. Minor errors do not prevent the server from starting up. However, the administrator must check any errors and verify that the mapping file available to the server is acceptable. Error messages from processing are logged or displayed based on the setting of the `_EUV_SVC_MSG_LOGGING` environment variable. See Table 6 on page 26 for more information on `_EUV_SVC_MSG_LOGGING`.

Note: If the Security server host is reconfigured, the Identity Mapping file must be reprocessed. You can reprocess the Identity Mapping file by calling the **RELOAD** function (see “Modifying and Deleting Identity Mapping Entries” on page 60). This is done because binding information from the security server is used when parsing the DCE identity value from the Identity Mapping file.

Step 3B: Set the `_ASU_IDMAP` Environment Variable: The `_ASU_IDMAP` environment variable must be set to the name of the Identity Mapping File used by the Application Support server. The declaration of this environment variable can be made in the `envar` file of the Application Support server. (For other ways of declaring environment variables, see *z/OS DCE Administration Guide*.) The format of the `_ASU_IDMAP` environment variable is:

```
_ASU_IDMAP=hfspathname
```

where *hfspathname* is the pathname of the Identity Mapping File. For example, if the HFS pathname of the Identity Mapping file is `/var/AS/server_1/idmap`, add the following entry to the `envar` file:

```
_ASU_IDMAP=/var/AS/server_1/idmap
```

You can choose to have one Identity Mapping File for each instance of the Application Support server, or a single Identity Mapping File for all instances of Application Support servers. In both cases, the `_ASU_IDMAP` environment variable can be set in the Application Support server's `envar` file.

Step 3C: Map Foreign Users to MVS User IDs:

Note: Perform this step only if there are Application Support server users from other cells (that is, foreign cells).

If there are foreign cells, the fully-qualified DCE name of each foreign principal must be mapped to an MVS user ID. To define this mapping, the foreign principals are entered in the Identity Mapping file. The foreign principal name must be translated to its corresponding UUID. Hence, it must read the mapping from the foreign cell. To be able to perform this, the administrator must have authority to read the `./:/sec/principal` directory in the foreign cell.

This is accomplished by having the DCE administrator add a *foreign_user* entry for the administrator in the ACL of the `./:/sec/principal` directory of the foreign cell. This entry must give this administrator read permission to this directory. For example, if the Application Support server belongs to the cell `./:/c=ca/o=ibm/ou=tid/cn=cell1`, and the principal name of the administrator is `cell_admin`, the ACL Editor session is as follows (performed on the foreign cell):

```
=>dcecp
dcecp> acl modify ./:/sec/principal -add { foreign_user \
./:/c=ca/o=ibm/ou=tid/cn=cell1/cell_admin r}
```

Modifying and Deleting Identity Mapping Entries

You may want to make changes to entries in the Identity Mapping file. To add, delete, or modify entries in the Application Support server and activate the changed Identity Mapping file, do the following:

- Step 1. Edit the Identity Mapping file to make the desired changes.
- Step 2. Call the **RELOAD** function from the ASUADMIN administration program or from the operator console by using the **MODIFY** system command.
- Step 3. Verify the RELOAD.

Note: Only use **RELOAD** if you are using the Identity Mapping file to map user IDs. If RACF is selected for determining the MVS user ID, the **RELOAD** function ends with a message indicating that **RELOAD** is not supported for RACF systems. See “Mapping DCE Principal Names to MVS User IDs with RACF” for more information on RACF support.

To call **RELOAD** from the ASUADMIN administration program, select the reload option from the Primary Options Panel. See Chapter 17, “Using the ASUADMIN Administration Program” on page 97.

You can also perform the **RELOAD** function:

- from the operator console by using the MODIFY system command. (see “Reloading the Identity Mapping File with the MODIFY Command” on page 116)
- with a customer-written client program using the vensrv_mgmt_reload_imt API (see “vensrv_mgmt_reload_imt” on page 132)

It is not necessary to restart the Application Support server to make changes take effect. The RELOAD will be successful even if there are minor errors in the Identity Mapping file. Any errors that are found in Identity Mapping file processing should be verified by the administrator to ensure that the Application Support server is running with an acceptable mapping file. Error messages from processing are logged or displayed based on the setting of the `_EUV_SVC_MSG_LOGGING` environment variable. See Table 6 on page 26 for more information on `_EUV_SVC_MSG_LOGGING`.

Access to the Identity Mapping File

The Identity Mapping file serves as the link between the DCE Security service and the local security subsystem. Access to the Identity Mapping file must be restricted to authorized administrators only.

The Application Support server must also be given read access to the Identity Mapping file that it uses. The administrator must be given read and write access to the Identity Mapping file.

Mapping DCE Principal Names to MVS User IDs with RACF

Although this book specifically references the z/OS DCE Security Server, the Application Support server can use any SAF product that supports the DCEUUIDS class profile and the R_dceruid callable service.

You may use RACF to perform the identity mapping function. If you already have RACF on your system, you may find advantages to using it for identity mapping rather than using the Identity Mapping file. The Identity Mapping file allows you to do specific mapping for a given server instance.

To map DCE principal names to MVS user IDs using RACF, perform the following steps:

- Step 3D: Set the `_ASU_EX_SEC` Environment Variable.
- Step 3E: Enable RACF services and authorize the Application Support server to use them.
- Step 3F: Map DCE principal names to MVS user IDs by creating DCEUUIDS class profiles.

Step 3D: Set the `_ASU_EX_SEC` Environment Variable: To map DCE principal names to MVS user IDs with RACF, you must set the `_ASU_EX_SEC` environment variable, described in Table 6 on page 26. To specify that you want to use RACF for identity mapping, be sure the following entry is in the `envar` file:

```
_ASU_EX_SEC=YES
```

Step 3E: Enable RACF Services and Authorize the Application Support Server: To map a DCE principal to an MVS user ID, the MVS user ID associated with the Application Support server must have read access to the profile `IRR.RDCERUID` in the `RACF FACILITY` class. **Assigning a UACC of `READ` to the profile `IRR.RDCERUID` is not recommended.**

The Application Support server uses RACF `DCEUUIDS` class profiles to map an MVS user ID to a DCE principal UUID. Before profiles defined to the `DCEUUIDS` class may be used, the class must be activated. To activate the `DCEUUIDS` class enter the following RACF command:

```
SETROPTS CLASSACT (DCEUUIDS)
```

Step 3F: Map DCE Principal Names to MVS User IDs through `DCEUUIDS` Class Profiles:

Profiles defined to the `DCEUUIDS` class are not expected to be administered via RACF commands by the RACF administrator. These profiles are created and maintained by using z/OS DCE `mvsexpt` and `mvsimpt` utilities. The `DCEUUIDS` class profiles are general RACF resource profiles. Although you can administer them using RACF `RDEFINE` and `RALTER` commands, it is strongly recommended that you use the `mvsexpt` and `mvsimpt` utilities, which are described in greater detail in the *z/OS DCE Administration Guide* and the *z/OS DCE Command Reference*.

Use the `mvsexpt` utility to define (or update) RACF principals that were previously defined to the DCE registry. One option of the `mvsexpt` utility allows you to create `DCEUUIDS` segments from an Application Support identity mapping file.

Use the `mvsimpt` command to define and modify DCE users that were previously defined to RACF.

RACF-DCE Interoperability Considerations for Application Support

Administrators: `DCEUUIDS` class profiles for use by the Application Support server must be defined in the following format:

```
cell_uuid.principal_uuid
```

When using the `mvsexpt` utility to create `DCEUUIDS` class profiles for the Application Support server, you **must** code your `EXPTVAR` file as **HOMEUUID YES**.

If you are using the `mvsexpt` utility to populate RACF entries from an Application Support Identity Mapping file, and want to implement full interoperability, remember:

- If your Identity Mapping file contains DCE principals from multiple DCE cells, you need to run the `mvsexpt` utility once per cell affected. The `CELLNAM` in the `EXPTVAR` must be modified to indicate in which cell the input principals reside, and you should limit the input to principals from that cell.
- If fully qualified global principal names such as `/.../dcecell7.endicott.ibm.com/zaccheus` are specified in your Identity Mapping file, you should either edit your Identity Mapping file to remove the cellname, or specify **HOMECELL NO** in your `EXPTVAR` control file.

Chapter 11. Configuring the CDS Namespace Entries

This chapter describes how you can configure CDS namespace entries for the Application Support server. To configure the CDS namespace entries for the Application Support server, you must:

1. Create the CDS directories that make up the pathname to the Application Support server object.
2. Give access to these namespace entries. The Application Support server administrator, the Application Support server clients, and the Application Support server principal require access rights to these entries.

Note: Although the DCE account of the Application Support server administrator has been created, this account does not yet have the required privileges to perform the tasks this section describes. The DCE administrator (sometimes called the cell administrator) must perform the procedures described here.

Creating CDS Directories for the Application Support Server

You can design the structure of the CDS namespace entry for the Application Support server in a manner that best suits your needs. The structure of the Application Support server namespace entry is determined by directories that you create as containers for the Application Support server objects.

The CDS directory entries to contain the Application Support server objects must first be created using the DCE administrative interface.

Note: Only directory entries that will contain the Application Support server object must be manually created using the DCE administrative interface. The object entry for the Application Support server is automatically created when you define the Application Support server, as discussed in Chapter 13, “Defining the Application Support Server to CDS” on page 73.

Depending on your requirements and preference, you can decide on either a simple or a lengthy pathname. For example, if you decide on a simple pathname, such as only one directory path to the Application Support server object, follow these steps:

1. Run the DCE administrative interface, `dcecp`. To start `dcecp` from TSO or from MVS, enter:

```
dcecp
```

2. At the (`dcecp>`) prompt, enter the following:

```
dcecp> directory create /./AS
```

Note: If you decide on a multi-level pathname for your CDS server directory entry, then you need to issue the `directory create` command for each level of the directory entry name you choose. For example, to create directory `/./AS/CICS`, make the following entries:

```
dcecp> directory create /./AS
```

```
dcecp> directory create /./AS/CICS
```

Access to the Application Support Server Namespace Entries

The Application Support server administrator, the Application Support server principal, and the DCE clients (who will access the Application Support server) must receive appropriate access privileges (also known as permissions) to the namespace entries of the Application Support server.

The Application Support server administrator requires the following permissions:

- Insert (i) permission on the directory that holds the Application Support server object entry.
- Read (r), write (w), and delete (d) permissions on the Application Support server object entry.

The Application Support server principal requires the read and write permissions on the Application Support server object entry.

The DCE clients accessing the Application Support server require the read permission on the Application Support server object entry.

You can specify the permissions to the directory and object namespace entries of the Application Support server using the ACL Editor. In a directory, the **Initial Object ACL** sets the ACL permissions for new object entries that are created in that directory.

Note that at this point, the object entry for the Application Support server has not yet been created. The Initial Object ACL will be used to set the ACL entries for the Application Support server object entry when it is created. (That is, when the Application Support server is defined to CDS, as discussed in Chapter 13, “Defining the Application Support Server to CDS” on page 73.)

For more information on the permissions CDS supports, see the DCE Directory Service part of *z/OS DCE Administration Guide*.

Procedure for Giving Access to the CDS Namespace Entries

Giving the administrator, the Application Support server principal, and the DCE clients appropriate permissions to the Application Support server namespace entry is done by running the ACL Editor on the specific Application Support server-related CDS entry.

To give the Application Support server administrator the required permission to the directory that will contain the Application Support server object entry, edit the ACL of the CDS directory using `dcecp acl` commands. For example, if the name of the directory is `./:/AS`, and the user ID of the administrator is `admin1`:

```
dcecp> acl modify ./:/AS -add {user admin1 i}
```

If this ACL already exists, you may need to issue the **-change** option instead of **-add**. See the *z/OS DCE Command Reference* for details.

To give the administrator, Application Support server principal, and DCE clients access to the Application Support server object, edit the ACL of the Initial Object of the `./:/AS` directory, and grant the following permissions:

- read, write, and delete permissions for the Application Support server administrator.
- read and write permissions for the Application Support server principal.
- read permission for the DCE clients who will access the Application Support server.

For example, for the administrator *admin1*, the Application Support server principal *p_server_1*, and the DCE clients of the Application Support server:

```
dcecp> acl modify ./AS -io -add {user admin1 rwd}
dcecp> acl modify ./AS -io -add {user p_server_1 rw}
dcecp> acl modify ./AS -io -add {any_other r}
```

If any of these ACLs already exist, you may need to issue the **-change** option instead of **-add**. See the *z/OS DCE Command Reference* for details.

The third entry for *any_other* gives DCE clients read permission to the Application Support server.

Note that all objects that are created inside this directory (including the Application Support server object entry) have this ACL.

If the Application Support Server Definition Has Been Deleted

When the definition of the Application Support server is deleted, the ACL to the Application Support server's namespace entry is also deleted. In this case, the definition of the ACL permissions to the Application Support server's namespace entry (as described in this section) must be performed again.

Creating Authorization Groups for Accessing the CDS Namespace

Another method of giving users access to the CDS namespace is to create a DCE Security group (using the DCE administrative interface, *dcecp*) which has all the permissions required to access the CDS namespace. Including the user as a member of this group automatically gives the user the required access rights to the objects in the CDS namespace. Creating DCE Security groups is discussed in the DCE Security Service part of *z/OS DCE Administration Guide*.

Chapter 12. Giving Access to the Administration Program

This chapter describes how to give the Application Support server administrator access to the administration program. The Access Control List of the Application Support server object controls access to the administration program. It also controls access to Encina **tkadmin** subcommands that you can use against an AS OTMA or AS CICS server; see page 135 for details.

To give the administrator access:

1. Create the initial ACL database files for the Application Support server.
2. Use the `_EUV_RPC_ACL_FILE` environment variable to specify the initial ACL database file.

Note: Although the DCE account of the Application Support server administrator has been created, this account does not yet have the required privileges to perform the tasks this section describes. The DCE administrator (sometimes called the cell administrator) must perform the procedures described here.

For a description of using the ACL Editor to give access to additional Application Support server administrators after the Application Support server has been initialized and started, see “Using the ACL Editor to Give Access to the Administration Program” on page 72.

Access to the Administration Program

The Application Support server administrator must be given the appropriate permissions to perform the following administrative tasks:

- Install and uninstall interfaces that the Application Support server supports.
- Start and stop attachment to these interfaces. To **attach** to the Application Support server means to enable the interfaces available in the Application Support server for a DCE client to call.
- Display the status of the servers and interfaces.
- List and test ACL entries.

Creating the initial ACL database for the Application Support server gives the administrator the appropriate permissions to the administration program. This database contains the initial entries (for example, an entry for the Application Support server administrator) of the Access Control List of the Application Support server.

Note: You must do this if the Application Support server is going to be started for the first time.

It is recommended that the ACL files reside in the same unique directory. The Application Support server must have read/write access to the ACL files.

To give permissions to **additional** Application Support server administrators, you can use the ACL Editor to modify the Access Control List associated with the Application Support server.

Note: You can use the ACL Editor to modify the ACL of the Application Support server (which controls access to the administration program) only if the Application Support server has been started and has finished initializing.

ACL Permissions Supported for the Administration Program

The following table lists and explains the permissions that are supported in accessing the Application Support server administration program:

Permission	Meaning
c	Control access. Allows a principal (or group) to modify the ACL database. Also gives the principal all the other permissions (r, l, i, x, t) mentioned in this table.
r	Read access. Allows a principal to display the status of the Application Support server and the interfaces that it supports. For tkadmin commands, you need read (r) permission for commands that do not change the target server's environment, such as list and query . See Chapter 20, "Using the tkadmin Command with the AS Server" on page 135 for information about tkadmin commands.
l	List access. Allows a principal to list the ACL entries.
i	Allows a principal to install and uninstall interfaces.
x	Allows a principal to start or stop attachment to the Application Support server. Also allows reloading of identity mapping file. For tkadmin commands, you need execute (x) permission for commands that could change the target server's environment, such as variations of set and trace . See Chapter 20, "Using the tkadmin Command with the AS Server" on page 135 for information about tkadmin commands.
t	Allows a principal to test the access rights to the Application Support server.

See Chapter 20, "Using the tkadmin Command with the AS Server" on page 135 for information about using these permissions with the **tkadmin** commands.

If there is more than one Application Support server administrator, you can vary the permissions given to each administrator, depending on the functions that they may perform, by editing the ACL of the Application Support server administration program.

Note: You must give at least one administrator Control (c) permission to edit the ACL. This is accomplished when the **ACL database file** is created at the time the Application Support server is started for the first time, as the following section describes.

Initial Access Control List

The Application Support server implements its own ACL Manager. The ACL Manager is that part of the server program that enforces access control on the server. The Application Support server ACL Manager uses an ACL database file to store the information on the access privileges that are given to the DCE users (including the Application Support server administrator).

Creating the Initial ACL Database Files

When the Application Support server is going to be started for the first time, you must create the following two files:

Version File	The Application Support server ACL Manager uses this file to switch between the current ACL database and the backup database.
ACL Database File	This file contains the initial ACL database of the Application Support server. This ACL database controls access to the administration program.

You can create these files using any desired pathname. Example ACL database files are in the **/usr/lpp/dceas/examples/common** directory. The file name of the ACL database is **aclsamp1.dat**. The file name of the version file is **aclsampv.dat**.

When you create the initial ACL database file, it is recommended that you use **oedit**. Do not use **acl_edit**. The entry in the initial ACL database file should not be a foreign principal.

Each instance of an Application Support server must have its own version and ACL database files.

Note: You must recreate the initial ACL database file every time the Security server host is reconfigured. When the Application Support server starts, it adds UUID information to the ACL database file. When the Security server host is reconfigured, this UUID information becomes obsolete.

ACL Database File Naming Convention

The version and ACL database files must use the following naming convention:

- Version file: *filenamev.dat*
- ACL database file: *filenameN.dat*

filename

Is any character string. This book refers to this as the **base** file name of the ACL database files. The base name must be the same for both the version and the ACL database files.

N Is either 1 or 2, depending on the version being used. When you create the initial ACL database file, you must use 1 as the value of *N* in the file name.

For example, if the *filename* is **as1acl**, the name of the version file is **as1aclv.dat**. The name of the ACL database file is **as1acl1.dat**.

Creating the Version File

Create one version file for each instance of the Application Support server. The initial entry (the only entry) in this file is **1**.

When the administrator later modifies and then saves to disk the ACL of the Application Support server, this file serves as a switch to internally indicate to the server which ACL database file to use.

Creating the ACL Database File

The ACL database file contains information on the number of ACL entries and the ACL entries themselves. Create this file using the following format:

number-of-entries

component entry-type permission cell-name principal-or-group-name
component entry-type permission cell-name principal-or-group-name
component entry-type permission cell-name principal-or-group-name

.

number-of-entries

Is an integer that corresponds to the number of ACL entries in the file.

component

Is the name of the server component, if the server maintains different ACL sets for the different components.

The Application Support server does not maintain different ACLs and, thus, the value of this field is always **NULL**.

<i>entry-type</i>	Can be any of the following: u user r foreign user g group p foreign group o any other f foreign other m mask n unauthenticated
<i>permission</i>	Can be one or more of the following: c control (all of the other permissions and the ability to modify the ACLs) l list the Access Control List r read x execute (start or stop attachment) and reload identity mapping file i insert (install or uninstall interfaces) t test access rights See “ACL Permissions Supported for the Administration Program” on page 68 for a discussion of these permissions. See Chapter 20, “Using the tkadmin Command with the AS Server” on page 135 and the Administrator's Guide for the Encina Toolkit for your system.
<i>cell-name</i>	Is the name or UUID of the foreign cell to which the principal or group belongs. This field is not required if the principal or group belongs to the local cell.
<i>principal-or-group-name</i>	Is the name or UUID of the principal or group.

Note: Do not edit the ACL database file after you have started the Application Support server for the first time. You must use the ACL Editor to modify the server's ACL.

Example ACL Database File

Figure 16 is an example of a sample ACL File.

```
2
NULL g c admin_group
NULL u rx operator1
```

Figure 16. Example ACL File

This example code gives the following permissions to the Application Support server:

- Control (c) permission for the group **admin_group**.
- Read (r) and execute (x) permission for the principal **operator1**.

Setting the `_EUV_RPC_ACL_FILE` Environment Variable

After creating the default ACL file, you must set the value of the `_EUV_RPC_ACL_FILE` environment variable to the **base** file name of the ACL database file, with the suffix **.dat**.

Each instance of the Application Support server must have its own ACL database file. Thus, the value of the `_EUV_RPC_ACL_FILE` environment variable for a specific Application Support server must be unique from other Application Support servers.

Note: The value set for this environment variable is neither the version file name (for example, **as1aclv.dat**), nor the ACL database file (for example, **as1acl1.dat**). Rather, it is the **base** file name that was the basis for the names of the version file and the ACL database file, suffixed with **.dat**. For example, it is simply **as1acl.dat**.

You can include the environment variable declaration in the `envar` file of the Application Support server's home directory. For example, if the ACL file is in the `/var/AS/server_1` directory, you would make the following declaration in the `envar` file:

```
_EUV_RPC_ACL_FILE=/var/AS/server_1/as1acl.dat
```

The Application Support server uses the value of this environment variable to locate the ACL database file.

See Chapter 4, “Creating the Environment Variable File” on page 21 for a discussion of the `envar` file.

Switching between Current and Backup Copies of the ACL File

You can use the version file to switch between the current ACL database file and the new ACL database file when you modify the current ACL file (using the ACL Editor). The following sequence of events explains how.

1. Before the first Application Support server startup, you create two files: the version file containing the entry 1 and the initial ACL database file, whose file name suffix is **1.dat**.
2. The Application Support server is started and becomes operational. The initial ACL database file is used to implement the permissions to the Application Support server.
3. The administrator modifies the ACL of the Application Support server (using the ACL Editor). At this point, the entry in the version file is automatically changed to 2. The modified ACL file is saved to a new ACL database file, whose suffix is **2.dat**. This becomes the basis for implementing the ACLs for the Application Support server. If the modified ACL file is saved successfully, the old ACL database file (whose file name suffix is **1.dat**) is deleted.

Note: This mechanism facilitates recovery if the modified ACL database file was not saved successfully. In this case, you can manually recover the previous ACL database file (the file with the suffix **1.dat**).

4. If the administrator again modifies the ACL of the Application Support server using the ACL Editor, the entry in the version file is switched back to 1. The modified ACL database file is saved to a file, the suffix of which is **1.dat**.
5. Each time the administrator modifies the ACL of the Application Support server, the entry in the version file is switched between 1 and 2. The suffix of the ACL database file name to which the modified ACL is saved is correspondingly switched between **1.dat** and **2.dat**.

Using the ACL Editor to Give Access to the Administration Program

Note: You cannot perform this procedure when the Application Support server is being configured and started for the first time. You can modify the ACL of the Application Support server using the ACL Editor only if the Application Support server has been started and has initialized. The procedure in this section describes how to give other administrators access to the administration program. You can do this only if you have already created the initial ACL database (as this section describes previously) and the Application Support server has been initialized and is running.

You can give other administrators access to the Application Support server by editing the ACL of the Application Support server object. You can add entries to the ACL either as:

- A user entry for the individual Application Support server administrator
- A group entry whose members are Application Support server administrators.

If the permissions are the same for all administrators, the second approach is recommended because it saves on administrative overhead. The following example illustrates this:

Creating and adding members to the administrator's group:

1. Use the DCE administrative interface from TSO or from MVS:

```
dcecp
```

2. Use the **group** subcommand to add the administrator's group, in this example, *admingrp*:

```
dcecp> group create admingrp
```

This assigns a default UNIX number and full name and includes the group, *admingrp*, in the project list. If you want to enter values other than the defaults, see the security part of the *z/OS DCE Administration Guide* for details.

3. Add the administrators to the group. In this example, two administrators are added: *admin1* and *admin2*.

```
dcecp> group add admingrp -member {admin1 admin2}
```

Editing the ACL:

4. Modify the ACL of the object representing the Application Support server, and give the administrator's group the appropriate permissions. The following example gives the group *admingrp* Control (c) permission to the Application Support server object *./AS/server_1*.

```
dcecp> acl modify -e ./AS/server_1 -add {group admingrp c }
```

Chapter 13. Defining the Application Support Server to CDS

Each instance of the Application Support server must be defined to the Cell Directory Service (CDS). To define the Application Support server to CDS, use the ASUADMIN administration program panels. This chapter summarizes how you can perform this task. Details on using the ASUADMIN program are described in Chapter 17, “Using the ASUADMIN Administration Program” on page 97.

Note: At this point, the account for the Application Support server administrator has been created and has been given the appropriate permissions to the administration program. The Application Support server administrator must perform the tasks this section describes.

What Does Defining to CDS Mean?

A CDS namespace entry must exist for the Application Support server so that DCE clients can locate this service. Certain attributes are entered into the CDS namespace entry of the Application Support server that defines the relationship between the Application Support server and the CICS or IMS subsystem to which it is attached.

Application Support's ASUADMIN administration program adds certain attributes to the CDS namespace entry that are specific to the type of server (CICS, APPC, OTMA, or ISC).

The Application Support server is administered through a hierarchy of ISPF panels displayed by the ASUADMIN administration program. Use the Create an IMS Application Support Server Definition and Create a CICS Application Support Server Definition panels to define the Application Support server to the DCE Cell Directory Service.

Defining the Application Support server to CDS means:

- Creating the CDS namespace entry for the Application Support server object.
- Adding the previously mentioned attributes of the Application Support server to the namespace entry.

Note: You can also modify the ACL of the individual Application Support server entry in the CDS namespace. However, you can do this only after the object entry for the Application Support server has been created in the CDS namespace. The object entry for the Application Support server is created when the Application Support server is defined using the administration panels. See “Steps in Defining the Application Support Server to CDS” on page 74 for details.

To modify ACL of the Application Support server object entry (do this only after the Application Support server has been defined to CDS):

1. Run the ACL Editor and use the **-e** option of the ACL Editor to modify the ACL of the Application Support server object entry in the CDS namespace. For example, for the Application Support server **./AS/server_1**:

```
acl_edit -e ./AS/server_1
```
2. Modify the ACL to give the administrator, the Application Support server principals, and the DCE clients the appropriate permissions (described previously) to the Application Support server namespace entry.

ASUADMIN Administration Program

The Application Support server provides an administration program, ASUADMIN, that is used to perform the daily administration tasks for Application Support server. One of the services that it provides is defining the Application Support server to CDS.

The ASUADMIN administration program provides panels that are used to enter information required for administration. Using these panels is described in more detail in Chapter 17, “Using the ASUADMIN Administration Program” on page 97.

Steps in Defining the Application Support Server to CDS

Important Note: Before you can use the ASUADMIN administration program, you must configure it. The procedure for configuring ASUADMIN is described in “Configuring the ASUADMIN Administration Program” on page 99.

To define the Application Support server to CDS, perform the following:

1. Login to DCE.
2. From ISPF, start the ASUADMIN administration program.

Note: Make sure that the administration program is configured properly to be run from TSO, as described in Chapter 17, “Using the ASUADMIN Administration Program” on page 97.

This displays the Primary Options Panel, shown in Figure 23 on page 100.

3. Select Option 5 of the Primary Options Panel. This displays the Work with Application Support Server Definitions panel, shown in Figure 28 on page 108.
4. Select the first option of this panel to create an Application Support server definition.
 - If you are creating a DCE AS CICS definition, enter 1 at the prompt for the subsystem.
 - If you are creating a DCE AS IMS with ISC definition, enter 2 at this prompt.
 - If you are creating a DCE AS IMS with APPC definition, enter 3 at this prompt.
 - If you are creating a DCE AS IMS with OTMA definition, enter 4 at this prompt.

For CICS and ISC, this displays the appropriate Create a CICS/IMS Application Support Server Definition panel. (For APPC and OTMA, the CDS entry is updated automatically, so no additional panel appears.)

The panel in Figure 17 on page 75 is displayed when defining an DCE AS CICS.

```

Create a CICS Application Support Server Definition

Server Name: ../AS/server_1

CICS Generic Application ID:  OETCICP4
Server Generic Application ID: CICSAS1

Command ==>
F1=Help   F2=Split  F3=Exit   F9=Swap   F12=Cancel

```

Figure 17. Create a CICS Application Support Server Definition Panel

The panel in Figure 18 is displayed when defining an DCE AS IMS with ISC.

```

Create an IMS Application Support Server Definition

Server Name  ../AS/server_2

IMS LU Name.....: IOEIMS4
Session Logon Mode Name.....: LU6NEGPS

Application Support Server LU Name.....: ASIMS
Application Support Server Device Prefix: IBMAPP5

Command ==>
F1=Help   F2=Split  F3=Exit   F9=Swap   F12=Cancel

```

Figure 18. Create an IMS ISC Application Support Server Definition Panel

Entering Application Support Server Attributes

For CICS and ISC only, you enter attributes in an additional panel. (For APPC and OTMA, the CDS entry is updated automatically.) The Application Support server attributes that are entered in the Define Server panel must be the same as the attributes that were entered when the CICS or IMS subsystem was configured.

See Chapter 5, “Configuring the AS CICS Server” on page 33 for configuration information for CICS and Chapter 8, “Configuring the AS IMS ISC Server” on page 47 for IMS with ISC.

For DCE AS CICS:

- The CICS Generic Application ID must match the APPLID defined for CICS.
- The Server Generic Application ID must be defined in the CICS Resource Definition. This must be the same as the NETNAME parameter in the CONNECTION definition.

For DCE AS IMS with ISC:

- The IMS LU name must match the VTAM name of IMS.
- The Session Logon Mode Name must match the label of the MODEENT macro (Mode name).
- The Application Support Server LU name must match the VTAM name of DCE AS IMS.
- The Application Support Server Device Prefix must match the subpool identifier prefix.

Chapter 14. Example Configuration Worksheets

This section provides example worksheets for configuring Application Support with CICS, IMS with APPC, IMS with OTMA, and IMS with ISC. The values shown on these worksheets are example values used in this book. Values appropriate to your system may be different. Chapter 15, “Blank Configuration Worksheets” on page 83 contains blank configuration worksheets you can use to fill in the configuration information you need when configuring your Application Support server.

AS CICS Worksheet

Table 9 is an example information worksheet for configuring AS CICS.

<i>Table 9 (Page 1 of 2). AS CICS Configuration Worksheet</i>	
Application Support Server Information	
Application Support server home directory	<code>/var/AS/server_1</code>
Application Support server envar file	<code>/var/AS/server_1/envar</code>
Application Support server CDS directory	<code>./AS</code>
Application Support server CDS entry name	<code>./AS/server_1</code>
Application Support server principal name	<code>p_server_1</code>
Keytab file pathname	<code>/var/AS/server_1/ktabfile</code>
ACL database version file	<code>/var/AS/server_1/as1aclv.dat</code>
ACL database file	<code>/var/AS/server_1/as1acl1.dat</code>
Environment Variable Values	
<code>_ASU_AUTOSTART</code>	<code>YES</code>
<code>_ASU_CICS_CONNECTIONS</code>	<code>20</code>
<code>_ASU_EX_SEC</code>	<code>NO</code>
<code>_ASU_IDMAP</code>	<code>/var/AS/server_1/idmap</code>
<code>_ASU_KEYTAB_FILE</code>	<code>/var/AS/server_1/ktabfile</code>
<code>_ASU_RESOURCE_MGR_NAME³</code>	<code>MVSSYS1CICSSERVER4</code>
<code>_ASU_TRAN³</code>	<code>YES</code>
<code>_EUV_RPC_ACL_FILE</code>	<code>/var/AS/server_1/as1acl.dat</code>
<code>_EUV_SVC_MSG_LEVEL</code>	<code>VERBOSE</code>
<code>_EUV_SVC_MSG_LOGGING</code>	<code>CONSOLE_LOGGING</code>
CICS Generic Application ID (APPLID)	<code>OETCICP4</code>
<code>ENCINA_CDS_ROOT³</code>	<code>./encina</code>
CICS CONNECTION Definitions	
ACCESSMETHOD	<code>IRC</code>
ATTACHSEC	<code>IDENTIFY</code>
CONNECTION	<code>AS1</code>
CONNTYPE	<code>SPECIFIC</code>
Server Generic Application ID (NETNAME)	<code>CICSAS1</code>
PROTOCOL	<code>EXCI</code>

³ This is for transactional RPCs only.

Table 9 (Page 2 of 2). AS CICS Configuration Worksheet

CICS SESSIONS Definitions	
CONNECTION	AS1
PROTOCOL	EXCI
RECEIVECOUNT	20
SESSIONS	SESSAS1

AS IMS with APPC Worksheet

Table 10 is an example information worksheet for configuring AS IMS with APPC.

<i>Table 10. AS IMS with APPC Configuration Worksheet</i>	
Application Support Server Information	
Application Support server home directory	<i>/var/AS/server_3</i>
Application Support server envar file	<i>/var/AS/server_3/envar</i>
Application Support server CDS directory	<i>./AS</i>
Application Support server CDS entry name	<i>./AS/server_3</i>
Application Support server principal name	<i>p_server_3</i>
Keytab file pathname	<i>/var/AS/server_3/ktabfile</i>
ACL database version file	<i>/var/AS/server_3/as3aclv.dat</i>
ACL database file	<i>/var/AS/server_3/as3acl1.dat</i>
Environment Variable Values	
<i>_ASU_APF</i>	<i>YES</i>
<i>_ASU_AUTOSTART</i>	<i>NO</i>
<i>_ASU_CONVERSATIONS</i>	<i>64</i>
<i>_ASU_EX_SEC</i>	<i>NO</i>
<i>_ASU_EXIT</i>	<i>AUTO</i>
<i>_ASU_EXPIRE</i>	<i>60</i>
<i>_ASU_IDMAP</i>	<i>/var/AS/server_3/idmap</i>
<i>_ASU_KEYTAB_FILE</i>	<i>/var/AS/server_3/ktabfile</i>
<i>_ASU_SERVER_LUNAME</i> (the same value as ACBNAME)	<i>MVSIMS02</i>
<i>_ASU_SYMBOLIC_NAME</i>	<i>IMS1</i>
<i>_EUV_RPC_ACL_FILE</i>	<i>/var/AS/server_3/as3acl.dat</i>
<i>_EUV_SVC_MSG_LEVEL</i>	<i>VERBOSE</i>
<i>_EUV_SVC_MSG_LOGGING</i>	<i>CONSOLE_LOGGING</i>

AS IMS with OTMA Worksheet

Table 11 is an example information worksheet for configuring AS IMS with OTMA.

<i>Table 11. AS IMS with OTMA Configuration Worksheet</i>	
Application Support Server Information	
Application Support server home directory	<code>/var/AS/server_4</code>
Application Support server envvar file	<code>/var/AS/server_4/envvar</code>
Application Support server CDS directory	<code>./AS</code>
Application Support server CDS entry name	<code>./AS/server_4</code>
Application Support server principal name	<code>p_server_4</code>
Keytab file pathname	<code>/var/AS/server_4/ktabfile</code>
ACL database version file	<code>/var/AS/server_4/as4aclv.dat</code>
ACL database file	<code>/var/AS/server_4/as4acl1.dat</code>
Environment Variable Values	
<code>_ASU_AUTOSTART</code>	<code>NO</code>
<code>_ASU_CONVERSATIONS</code>	<code>64</code>
<code>_ASU_EX_SEC</code>	<code>NO</code>
<code>_ASU_EXPIRE</code>	<code>60</code>
<code>_ASU_IDMAP</code>	<code>/var/AS/server_4/idmap</code>
<code>_ASU_KEYTAB_FILE</code>	<code>/var/AS/server_4/ktabfile</code>
<code>_ASU_RESOURCE_MGR_NAME³</code>	<code>MVSSYS1IMSSYS1SERVER4</code>
<code>_ASU_TRAN³</code>	<code>YES</code>
<code>_ASU_XCF_GROUP_NAME</code>	<code>MVSSYS1</code>
<code>_ASU_XCF_IMS_MEMBER_NAME</code>	<code>IMSSYS1</code>
<code>_ASU_XCF_SERVER_MEMBER_NAME</code>	<code>SERVER4</code>
<code>_EUV_RPC_ACL_FILE</code>	<code>/var/AS/server_4/as4acl.dat</code>
<code>_EUV_SVC_MSG_LEVEL</code>	<code>VERBOSE</code>
<code>_EUV_SVC_MSG_LOGGING</code>	<code>CONSOLE_LOGGING</code>
<code>ENCINA_CDS_ROOT³</code>	<code>./encina</code>

AS IMS with ISC Worksheet

Table 12 is an example information worksheet for configuring AS IMS with ISC.

<i>Table 12. AS IMS with ISC Configuration Worksheet</i>	
Application Support Server Information	
Application Support server home directory	<i>/var/AS/server_2</i>
Application Support server envvar file	<i>/var/AS/server_2/envvar</i>
Application Support server CDS directory	<i>./AS</i>
Application Support server CDS entry name	<i>./AS/server_2</i>
Application Support server principal name	p_server_2
Keytab file pathname	<i>/var/AS/server_2/ktabfile</i>
ACL database version file	<i>/var/AS/server_2/as2aclv.dat</i>
ACL database file	<i>/var/AS/server_2/as2acl1.dat</i>
Environment Variable Values	
_ASU_AUTOSTART	YES
_ASU_EX_SEC	NO
_ASU_IDMAP	<i>/var/AS/server_2/idmap</i>
_ASU_KEYTAB_FILE	<i>/var/AS/server_2/ktabfile</i>
_EUV_RPC_ACL_FILE	<i>/var/AS/server_2/as2acl.dat</i>
_EUV_SVC_MSG_LEVEL	VERBOSE
_EUV_SVC_MSG_LOGGING	CONSOLE_LOGGING
IMS Information	
IMS LU Name (ACBNAME)	IOEIMS4
Session Logon Mode Name	LU6NEGPS
Application Support server LU Name (ACBNAME)	ASIMS
Application Support server Device Prefix	IBMAPPS
Number of Sessions	5
Session Logon Password	OUR1PASS

Chapter 15. Blank Configuration Worksheets

This section provides blank worksheets for configuring AS CICS, IMS with APPC, IMS with OTMA, and AS IMS with ISC. Fill in the values here and use the worksheet when configuring your Application Support server.

AS CICS Worksheet

Table 13 is a blank information worksheet for configuring AS CICS. It includes values for attributes that can have only one value.

<i>Table 13 (Page 1 of 2). AS CICS Configuration Worksheet</i>	
Application Support Server Information	
Application Support server home directory	
Application Support server envar file	
Application Support server CDS directory	
Application Support server CDS entry name	
Application Support server principal name	
Keytab file pathname	
ACL database version file	
ACL database file	
Environment Variable Values	
_ASU_AUTOSTART	
_ASU_CICS_CONNECTIONS	
_ASU_EX_SEC	
_ASU_IDMAP	
_ASU_KEYTAB_FILE	
_ASU_RESOURCE_MGR_NAME ⁴	
_ASU_TRAN ⁴	
_EUV_RPC_ACL_FILE	
_EUV_SVC_MSG_LEVEL	
_EUV_SVC_MSG_LOGGING	
ENCINA_CDS_ROOT ⁴	
CICS Generic Application ID (APPLID)	
CICS CONNECTION Definitions	
ACCESSMETHOD	IRC
ATTACHSEC	IDENTIFY
CONNECTION	
CONNTYPE	SPECIFIC
Server Generic Application ID (NETNAME)	
PROTOCOL	EXCI
CICS SESSIONS Definitions	

⁴ This is for transactional RPCs only.

<i>Table 13 (Page 2 of 2). AS CICS Configuration Worksheet</i>	
CONNECTION	
PROTOCOL	EXCI
RECEIVECOUNT	
SESSIONS	

AS IMS with APPC Worksheet

Table 14 is a blank information worksheet for configuring AS IMS with APPC.

<i>Table 14. AS IMS with APPC Configuration Worksheet</i>	
Application Support Server Information	
Application Support server home directory	
Application Support server envvar file	
Application Support server CDS directory	
Application Support server CDS entry name	
Application Support server principal name	
Keytab file pathname	
ACL database version file	
ACL database file	
Environment Variable Values	
_ASU_APF	
_ASU_AUTOSTART	
_ASU_CONVERSATIONS	
_ASU_EX_SEC	
_ASU_EXIT	
_ASU_EXPIRE	
_ASU_IDMAP	
_ASU_KEYTAB_FILE	
_ASU_SERVER_LUNAME (the same value as ACBNAME)	
_ASU_SYMBOLIC_NAME	
_EUV_RPC_ACL_FILE	
_EUV_SVC_MSG_LEVEL	
_EUV_SVC_MSG_LOGGING	

AS IMS with OTMA Worksheet

Table 15 is a blank information worksheet for configuring AS IMS with OTMA.

<i>Table 15. AS IMS with OTMA Configuration Worksheet</i>	
Application Support Server Information	
Application Support server home directory	
Application Support server envvar file	
Application Support server CDS directory	
Application Support server CDS entry name	
Application Support server principal name	
Keytab file pathname	
ACL database version file	
ACL database file	
Environment Variable Values	
_ASU_AUTOSTART	
_ASU_CONVERSATIONS	
_ASU_EX_SEC	
_ASU_EXPIRE	
_ASU_IDMAP	
_ASU_KEYTAB_FILE	
_ASU_RESOURCE_MGR_NAME ⁴	
_ASU_TRAN ⁴	
_ASU_XCF_GROUP_NAME	
_ASU_XCF_IMS_MEMBER_NAME	
_ASU_XCF_SERVER_MEMBER_NAME	
_EUV_RPC_ACL_FILE	
_EUV_SVC_MSG_LEVEL	
_EUV_SVC_MSG_LOGGING	
ENCINA_CDS_ROOT ⁴	

AS IMS with ISC Worksheet

Table 16 is a blank information worksheet for configuring AS IMS with ISC.

<i>Table 16. AS IMS with ISC Configuration Worksheet</i>	
Application Support Server Information	
Application Support server home directory	
Application Support server envvar file	
Application Support server CDS directory	
Application Support server CDS entry name	
Application Support server principal name	
Keytab file pathname	
ACL database version file	
ACL database file	
Environment Variable Values	
_ASU_AUTOSTART	
_ASU_EX_SEC	
_ASU_IDMAP	
_ASU_KEYTAB_FILE	
_EUV_RPC_ACL_FILE	
_EUV_SVC_MSG_LEVEL	
_EUV_SVC_MSG_LOGGING	
IMS Information	
IMS LU Name (ACBNAME)	
Session Logon Mode Name	
Application Support server LU Name (ACBNAME)	
Application Support server Device Prefix	
Number of Sessions	
Session Logon Password	

Chapter 16. Starting the Application Support Server

This chapter describes some of the features of PROCs that are used to start the Application Support server.

Startup Procedure for DCE AS CICS

Figure 19 on page 90 is an example procedure to start DCE AS CICS. In the JCL that calls the procedure, you must ensure that:

- The `_EUV_HOME` variable is set to the pathname where the `envar` file is read, for example:
`/var/AS/server_1`
- The `SYSIN` definition statement points to a data set that has two lines: the CDS namespace entry of the Application Support server object and the DCE principal identity of the Application Support server, for example:

```
././AS/server_1  
p_server_1
```

Note: You can use a pre-allocated data set as input (with the `DSNAME` parameter) containing the lines in the preceding example, rather than coding them in the JCL. If you do, be sure that you do **not** include line sequence numbers in it. Sequence numbers are interpreted as part of the CDS namespace entry name or the principal DCE identity. This causes an error when you try to start the Application Support server.

In the startup PROC:

- The `STEPLIB` includes the stub library. `SDFHEXCI` is the CICS EXCI load module library.
- Change `CICSPFX` to the location of CICS on your system.

```

//*****
//* ASUVSC - startup procedure for the DCE to CICS AS server
//*****
//*****
//* Sample JCL to invoke the procedure
//*****
//* //ASUVCICS EXEC ASUVSC,
//* // XSTUBLIB='TSDCEPR.CICS.AUTHLIB',
//* // CICSSPFX='SHARE.CICS410',
//* // PARM=('ENVAR('_EUV_HOME=/var/AS/server_1')/')
//* //GO.SYSIN DD *
//* ./AS/server_1
//* p_server_1
//* /*
//*
//* SYSIN contains:
//*           Server CDS name
//*           Server DCE principal name
//*****
//ASUVSC PROC XSTUBLIB=,REGSIZE=0M,PARMS=,
// OUTCLASS='*',
// CICSSPFX='CICS.V4R1'
//GO EXEC PGM=ASUVSC,REGION=&REGSIZE.,TIME=1440,PARM=&PARMS
//*****
//* LOAD LIBRARIES
//*****
//STEPLIB DD DSN=&XSTUBLIB,DISP=SHR
// DD DSN=&CICSSPFX..SDFHEXCI,DISP=SHR
//*****
//SYSIN DD DUMMY
//SYSOUT DD SYSOUT=&OUTCLASS
//SYSPRINT DD SYSOUT=&OUTCLASS
//CEEDUMP DD SYSOUT=&OUTCLASS
//SYSUDUMP DD SYSOUT=&OUTCLASS

```

Figure 19. Example Procedure to Start DCE AS CICS

Application Support provides an example PROC, member ASUVSC in the *ASPREFIX.SASUSAMP* data set.

Startup Procedure for DCE AS IMS with OTMA

Figure 20 is an example procedure to start DCE AS IMS with OTMA. In the JCL that calls this procedure, you must ensure that:

- The `_EUV_HOME` variable is set to the pathname where the envar file will be read, for example, `/var/AS/server_4`.
- The SYSIN definition statement points to a data set that has two lines: the CDS namespace entry of the Application Support server and the DCE principal identity of the Application Support server. For example:

```

./AS/server_4
p_server_4

```

Note: You may use a pre-allocated data set as input (with the DSNAME parameter) containing the lines in the preceding example, rather than coding them in the JCL. If you do, be sure that you do **not** include line sequence numbers in it. Sequence numbers are interpreted as part of the CDS namespace entry name or the principal DCE identity. This causes an error when you try to start the Application Support server.

In the startup PROC, the STEPLIB includes the stub library.

```

//*****
/* ASUVSO - startup procedure for the DCE to IMS OTMA AS server
//*****
//*****
/* Sample JCL to invoke the procedure
//*****
/* //ASUVIMS EXEC ASUVSO,
/* // XSTUBLIB='TSDCEPR.IMS.STUBLIB',
/* // PARM=('ENVAR('_EUV_HOME=/var/AS/server_4')/')
/* //GO.SYSIN DD *
/* ./AS/server_4
/* p_server_4
/* /*
/*
/* SYSIN contains:
/*           Server CDS name
/*           Server DCE principal name
/*
//ASUVSO PROC XSTUBLIB=,REGSIZE=0M,PARMS=,
// OUTCLASS='*'
//-----
//GO EXEC PGM=ASUVSO,REGION=&REGSIZE.,TIME=1440,PARM=&PARMS
//*****
/* LOAD LIBRARIES
//*****
//STEPLIB DD DSN=XSTUBLIB,DISP=SHR
//*****
//SYSIN DD DUMMY
//*****
//SYSOUT DD SYSOUT=&OUTCLASS
//SYSRINT DD SYSOUT=&OUTCLASS
//CEEDUMP DD SYSOUT=&OUTCLASS
//SYSUDUMP DD SYSOUT=&OUTCLASS

```

Figure 20. Example Procedure to Start DCE AS IMS with OTMA

Application Support provides an example PROC, member ASUVSO in the *ASPREFX.SASUSAMP* data set.

Startup Procedure for DCE AS IMS with APPC

Figure 21 is an example procedure to startup DCE AS IMS with APPC. In the JCL that calls this procedure, you must ensure that:

- The `_EUV_HOME` variable is set to the pathname where the `envar` file will be read, for example, `/var/AS/server_3`.
- The `SYSIN` definition statement points to a data set that has two lines: the CDS namespace entry of the Application Support server and the DCE principal identity of the Application Support server. For example:

```

./AS/server_3
p_server_3

```

Note: You may use a pre-allocated data set as input (with the `DSNAME` parameter) containing the lines in the preceding example, rather than coding them in the JCL. If you do, be sure that you do **not** include line sequence numbers in it. Sequence numbers are interpreted as part of the CDS namespace entry name or the principal DCE identity. This causes an error when you try to start the Application Support server.

In the startup PROC, the `STEPLIB` includes the stub library.

```

//*****
//* ASUVSA - startup procedure for the DCE to IMS APPC AS server
//*****
//*****
//* Sample JCL to invoke the procedure
//*****
//* //ASUVIMS EXEC ASUVSA,
//* // XSTUBLIB='TSDCEPR.IMS.STUBLIB',
//* // PARS=('ENVAR('_EUV_HOME=/var/AS/server_3')/')
//* //GO.SYSIN DD *
//* ./AS/server_3
//* p_server_3
//* /*
//*
//* SYSIN contains:
//*           Server CDS name
//*           Server DCE principal name
//*
//ASUVSA  PROC XSTUBLIB=,REGSIZE=0M,PARMS=,
// OUTCLASS='*'
//-----
//GO      EXEC PGM=ASUVSA,REGION=&REGSIZE.,TIME=1440,PARM=&PARMS
//*****
//* LOAD LIBRARIES
//*****
//STEPLIB DD DSN=&XSTUBLIB,DISP=SHR
//*****
//SYSIN   DD DUMMY
//*****
//SYSOUT  DD SYSOUT=&OUTCLASS
//SYSPRINT DD SYSOUT=&OUTCLASS
//CEEDUMP DD SYSOUT=&OUTCLASS
//SYSUDUMP DD SYSOUT=&OUTCLASS

```

Figure 21. Example Procedure to Start DCE AS IMS with APPC

Application Support provides an example PROC, member ASUVSA in the *ASMPREFX.SASUSAMP* data set.

Startup Procedure for DCE AS IMS with ISC

Figure 22 is an example procedure to start DCE AS IMS with ISC. In the JCL that calls this procedure, you must ensure that:

- The `_EUV_HOME` variable is set to the pathname where the `envar` file will be read, for example, `/var/AS/server_2`.
- The `SYSIN` definition statement points to a data set that has three lines: the CDS namespace entry of the Application Support server, the DCE principal identity of the Application Support server, and the VTAM password for the Application Support server LU Name. This must be the same as the `PRTCT` VTAM definition, as described in “Step 2: Specify VTAM Definitions in the VTAM Definition Library” on page 49. For example:

```

./AS/server_2
p_server_2
OUR1PASS

```

Note: You may use a pre-allocated data set as input (with the `DSNAME` parameter) containing the lines in the preceding example, rather than coding them in the JCL. If you do, be sure that you do **not** include line sequence numbers in it. Sequence numbers are interpreted as being part of the CDS namespace entry name or the principal DCE identity. This causes an error when you try to start the Application Support server.

In the startup PROC, the STEPLIB includes the stub library.

```
//*****
//* ASUVSI - startup procedure for the DCE to IMS ISC AS server
//*****
//*****
//* Sample JCL to invoke the procedure
//*****
//* //ASUVIMS EXEC ASUVSI,
//* // XSTUBLIB='TSDCEPR.IMS.STUBLIB',
//* // PARM=('ENVAR('_EUV_HOME=/var/AS/server_2')/')
//* //GO.SYSIN DD *
//* ./AS/server_2
//* p_server_2
//* OURIPASS
//* /*
//* //GO.ASUVPLOG DD SYSOUT=*
//*
//* SYSIN contains:
//*          Server CDS name
//*          Server DCE principal name
//*          VTAM Password to Server's ACB
//*
//ASUVSI PROC XSTUBLIB=,REGSIZE=0M,PARMS=,
// OUTCLASS='*'
//-----
//GO EXEC PGM=ASUVSI,REGION=&REGSIZE.,TIME=1440,PARM=&PARMS
//*****
//* LOAD LIBRARIES
//*****
//STEPLIB DD DSN=&XSTUBLIB,DISP=SHR
//*****
//SYSIN DD DUMMY
//*****
//ASUVPLOG DD SYSOUT=&OUTCLASS
//SYSOUT DD SYSOUT=&OUTCLASS
//SYSPRINT DD SYSOUT=&OUTCLASS
//CEEDUMP DD SYSOUT=&OUTCLASS
//SYSUDUMP DD SYSOUT=&OUTCLASS
```

Figure 22. Example Procedure to Start DCE AS IMS with ISC

The ASUVPLOG DD statement is explained in Chapter 22, “Problem Determination” on page 155. Application Support provides an example PROC, member ASUVSI in the *ASPREFX.SASUSAMP* data set.

Part 3. Administering the Application Support Server

This part of the book describes:

- The ASUADMIN administration program that you use to administer the Application Support server.
- Administering the Application Support server from the operator console
- Administering the Application Support server with a client program
- Using **tkadmin** commands with the Application Support server
- Managing AS IMS or CICS server transactions called with TRPC
- Problem determination

Chapter 17. Using the ASUADMIN

Administration Program	97
Administration Panel Functions	98
Configuring the ASUADMIN Administration Program	99
Starting the Administration Program	99
Primary Options Panel	99
Messages from the Administration Panels	100
CDS Names	101
Server Name	102
Attaching the Application Support Server	102
Limitations of Attaching the DCE AS IMS Server	102
Stopping an Application Support Server Attachment	103
Stopping and Restarting Attachment of DCE AS CICS Numerous Times	103
Displaying Interfaces and AS Server Operational Data	103
Installing and Uninstalling Specific Interfaces	103
Prerequisites to Interface Installation	104
Action Codes	105
Interface Name	105
Response Field	105
Installed Interfaces That Have Been Deleted	106
Installing and Uninstalling Interfaces in Batch	107
Working with Application Support Server Definitions	108
Defining an Application Support Server	108
Displaying Data About an Application Support Server	111
Modifying Data about an Application Support Server	111
Deleting the Definition of an Application Support Server	111
Logging In to z/OS DCE	112
Reloading the Identity Mapping File	113

Chapter 18. Administering the AS Server

Using MVS System Commands	115
Starting Attachment of an Application Support Server	115

Stopping Attachment of an Application Support Server	116
Reloading the Identity Mapping File with the MODIFY Command	116
Displaying the Server's Operational Status	117
Displaying the Status of Interfaces	119
Installing and Uninstalling Interfaces	120
Stopping the Application Support Server	120

Chapter 19. Administering the AS Server

Using a Client Program	123
What You Need to Build the Client	123
vensrv_mgmt_display_if	124
Description	124
Format	124
Parameters	124
Return Values	125
vensrv_mgmt_display_ifs	126
Description	126
Format	126
Parameters	126
Return Values	126
vensrv_mgmt_display_server	128
Description	128
Format	128
Parameters	128
Return Values	129
vensrv_mgmt_if_mgmt	130
Description	130
Format	130
Parameters	130
Return Values	131
vensrv_mgmt_reload_imt	132
Description	132
Format	132
Parameters	132
Return Values	132
vensrv_mgmt_start_attachment	133
Description	133
Format	133
Parameters	133
Return Values	133
vensrv_mgmt_stop_attachment	134

Description	134	Transactions and Locking in IMS	144
Format	134	Viewing AS Server Encina Transaction	
Parameters	134	Information	144
Return Values	134	Administering Transactions	146
		Intervening in Problematic Transactions	147
		Cold Log Starts	152
Chapter 20. Using the tkadmin Command			
with the AS Server	135		
Chapter 21. Managing Transactions		Chapter 22. Problem Determination	155
Called with Transactional RPCs	137	Setting _EUV_SVC_MSG_LEVEL	155
Background on Encina Transaction		Setting _EUV_SVC_MSG_LOGGING	156
Processing	139	Error Correlator ID	156
The Two-Phase Commit Protocol	139	Using Message ASUV430I for Problem	
The States of an AS IMS or AS CICS		Solving in AS IMS	156
Encina Transaction	140	Using the ASUVPLOG Data Set to Capture	
Transaction Identifiers	141	IMS Messages	156
		Allocating the ASUVPLOG Data Set	157

Chapter 17. Using the ASUADMIN Administration Program

There are several ways to perform many administrative tasks:

- Use the ASUADMIN administration program
- Use MVS System Commands
- Use a customer-written client program that includes administrative APIs

The following table lists various administrative tasks references information on the different ways of performing these tasks.

<i>Table 17 (Page 1 of 2). Administrative Tasks and Ways to Do Them</i>			
Task	ASUADMIN	MODIFY Command	Administration API
Start attachment of an Application Support server	See "Attaching the Application Support Server" on page 102	See "Starting Attachment of an Application Support Server" on page 115	See "vensrv_mgmt_start_attachment" on page 133
Stop attachment of an Application Support server	See "Stopping an Application Support Server Attachment" on page 103	See "Stopping Attachment of an Application Support Server" on page 116	See "vensrv_mgmt_stop_attachment" on page 134
Reload the identity mapping file	See "Reloading the Identity Mapping File" on page 113	See "Reloading the Identity Mapping File with the MODIFY Command" on page 116	See "vensrv_mgmt_reload_imt" on page 132
Query a server's operational status	See "Displaying Interfaces and AS Server Operational Data" on page 103	See "Displaying the Server's Operational Status" on page 117	See "vensrv_mgmt_display_server" on page 128
Query interfaces installed on an Application Support server	See "Displaying Interfaces and AS Server Operational Data" on page 103	See "Displaying the Status of Interfaces" on page 119	See "vensrv_mgmt_display_ifs" on page 126
Display the status of an interface	See "Displaying Interfaces and AS Server Operational Data" on page 103	See "Displaying the Status of Interfaces" on page 119	See "vensrv_mgmt_display_if" on page 124

Table 17 (Page 2 of 2). Administrative Tasks and Ways to Do Them

Task	ASUADMIN	MODIFY Command	Administration API
Install or uninstalling an interface	“Installing and Uninstalling Specific Interfaces” on page 103	See “Installing and Uninstalling Interfaces” on page 120	See “vensrv_mgmt_if_mgmt” on page 130
Stopping the Application Support Server	N/A	See “Stopping the Application Support Server” on page 120	N/A

This chapter describes the use of the ASUADMIN administration program and the administration panels that running this program displays. This chapter also describes the **ASUVIFM** utility that installs or uninstalls interfaces in batch.

Administration Panel Functions

Application Support server provides ISPF panels that you use to perform the following tasks:

- Attach an Application Support server to a CICS or IMS subsystem.
Attaching an Application Support server to a CICS or IMS subsystem enables the Application Support server to accept client requests for CICS or IMS transactions.
- Stop the attachment of the Application Support server from a CICS or IMS subsystem.
A stop attach request disables the Application Support server from accepting client calls to CICS or IMS transactions.
- Display interface and operational data.
You can view information on the state of the Application Support server and its supported interfaces.
- Install and uninstall interfaces supported by the Application Support server.
Installing an interface advertises its availability to DCE clients through the DCE Directory Service. Uninstalling an interface makes it unavailable to clients.
- Define, delete, display, or modify an instance of an Application Support server to the Cell Directory Service.
When you define the Application Support server to CDS, essential attributes are entered to the CDS namespace that are used by the Application Support server in establishing communication links with a specific CICS or IMS region. You can also delete an instance of an Application Support server or display and modify data about a specific Application Support server.
- Login to DCE.
To perform administrative functions on the Application Support server, you must be logged in to DCE. You can log in to DCE while running the administration program by using an administration panel. You can also use this function if you want to use a different DCE userid than the one that you are currently logged in when administering the Application Support server.
- Reload the Identity Mapping file.
If you are using the Identity Mapping file, and you make changes to it, you must reload it to make the changes take effect.

Note: The tasks described in this section must be performed by the Application Support server administrator.

Configuring the ASUADMIN Administration Program

Before running the ASUADMIN administration program, perform the following configuration steps:

- Add the SASUPxxx data set to your ISPPLIB concatenation, where xxx is the language identifier of the language you want the administration panels to be displayed in. Specify ENU for American English and JPN for Japanese.
- Add the SASUMxxx data set to your ISPMLIB concatenation, where xxx is the language identifier of the language you want the administration panels to be displayed in. Specify ENU for American English and JPN for Japanese.
- Add the SASUTLIB library to your ISPTLIB concatenation.
- Add the SASUEXEC library to your SYSEXEC or SYSPROC concatenation.
- Be sure that the environment variable file contains the NLSPATH variable. NLSPATH should indicate the pathname where your Application Support message catalog (dceven.cat) is located. See Table 6 on page 26 for more information on NLSPATH.

The environment variable file should be in your home directory in HFS.

Starting the Administration Program

To start the Application Support server administration program, enter **ASUADMIN** from TSO within a valid ISPF environment.

Note: To start the administration program from the TSO READY prompt, enter the following:

```
ISPF PGM(ASUV) NEWAPPL(ASUV)
```

Primary Options Panel

The primary options panel contains several selections representing the primary functions that are available for administering the Application Support server. This panel serves as an entry point to a hierarchical flow of subpanels used for performing the administrative tasks.

The primary options panel shown in Figure 23 on page 100 is displayed.

```
Application Support Server Administration
Command ==>
Type an Application Support Server name. Select an option. Then press Enter.
Server Name: /./AS/server_1

- 1. Attach Application Support Server to Subsystem
  2. Detach Application Support Server from Subsystem
  3. Display Interfaces and Application Support Server Operational Data
  4. Install / Uninstall Specific Interfaces
  5. Work with Application Support Server Definitions
  6. Perform DCE Login
  7. Reload Identity Mapping File

OS/390 DCE Application Support
Licensed Materials - Property of IBM
5647-A01 (C) Copyright IBM Corporation 1994, 1999. All rights reserved.

F1=Help  F2=Split  F3=Exit  F9=Swap  F12=Cancel
```

Figure 23. Primary Options Panel

Notes:

1. In ISPF 4.1, the user can specify KEYLIST to be ON or OFF. If KEYLIST is OFF, ISPF ignores the PF key definitions that were made on the administration panels. Instead, it displays the system PF keys. The system PF key definitions are not compatible with the PF keys on the administration panels. To run the administration panels with the proper PF keys, KEYLIST must be set to ON. This is the default setting.
2. If you select option 3 (Display Interfaces and Application Support Server Operational Data), the next panel includes an option to Display IMS Session Information only if the server is AS IMS with ISC. (This option does not appear for AS CICS or AS IMS with APPC or OTMA.)

Messages from the Administration Panels

The administration panels display messages that may be informational, warnings, or those that request a specific action. These messages appear as pop-up windows across the panels. Figure 24 on page 101 shows the message displayed by the primary options panel after selecting Option 1.

Press Enter or Cancel (F12) to acknowledge these messages and continue working with the panels.

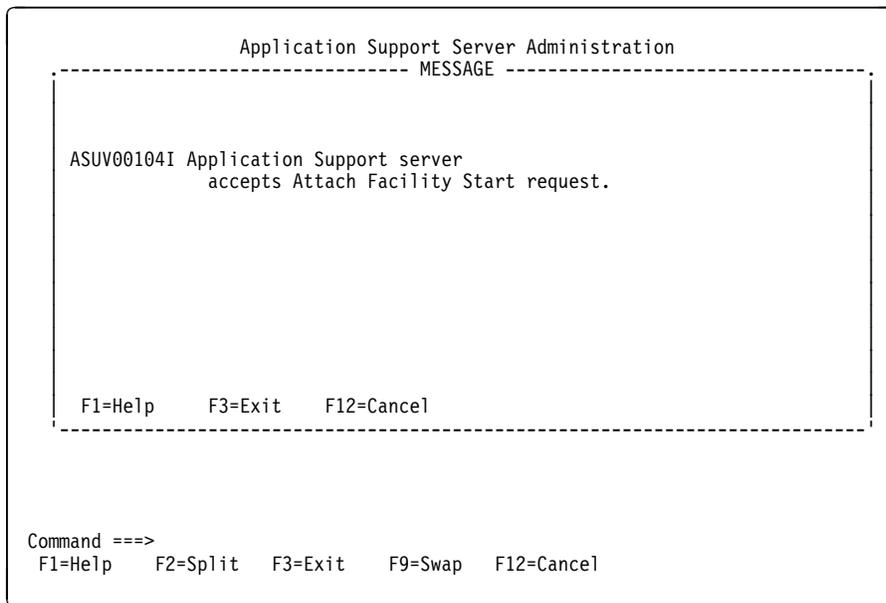


Figure 24. Messages on Panels

CDS Names

Every cell contains at least one system running a CDS server. A CDS server stores and maintains names and handles requests to create, modify, and look up data. The total collection of names shared by CDS servers in a cell is called the cell namespace. The cell namespace administrator can organize CDS names into a hierarchical structure of directories. CDS directories, conceptually similar to the directories in hierarchical file systems of many operating systems (for example, UNIX), are a logical way to group names for ease of management and use.

In a cell namespace, any directory that has a directory beneath it is considered the parent of the directory below it. Any directory that has a directory above it is considered a child of the directory above it. The top level of the cell namespace is called the **cell root**. You can refer to the cell root either by the global name of the cell (*././cell-name*) or by the short-form *./.* prefix.

Figure 25 shows a simple cell namespace hierarchy, starting at the cell root. The cell root (*./.*) is the parent of the directories named *././hosts* and *././subsys*. The *././subsys* directory is a child of the cell root directory and the parent of the *././subsys/dce* directory.

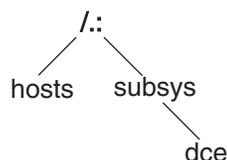


Figure 25. Sample CDS Namespace Hierarchy

The complete specification of a CDS name, going from left to right from the cell root to the entry being named, is called the CDS full name. Each element within a full name is separated by a slash (*/*) and is called a simple name. For example, suppose the *././hosts* directory shown in the preceding figure contains an entry for a host whose simple name is **bargle**. The CDS full name of that entry is ***././hosts/bargle***. For more details on CDS names, see the DCE Directory Service part of *z/OS DCE Administration Guide*.

Server Name

All selections on the primary options panel must be directed to a specific instance of an Application Support server by specifying the CDS name of the Application Support server. You must fill the **Server Name** field in the primary options panel with the appropriate name corresponding to the specific Application Support server (see Figure 23 on page 100).

In DCE, a server is assigned a CDS name, such as `./:AS/server_1`. You must enter the CDS full name of the Application Support server in the Server Name field of the primary options panel. The Server Name in the administration panels cannot be more than 256 characters.

When the administration panels are started, the Server Name field displays the name that was entered on the field during the last call to the administration panels. If you need to change the Server Name, you can either reenter the entire name or edit the appropriate portions of the displayed name. The validity of the server name entered in the primary options panel is checked only after a specific DCE action is requested regarding that Application Support server.

Attaching the Application Support Server

After an Application Support server has been started and initialized, it still cannot accept remote procedure calls to CICS or IMS transaction programs. Selecting option 1, Attach Application Support Server to Subsystem in the primary options panel (see Figure 23 on page 100) enables the instance of the Application Support Server (as defined in the Server Name field) to be in a state in which it can accept remote procedure calls to CICS or IMS transactions that it supports. When this is selected, the state of the Application Support server is changed from quiescent to operational. Server states are described in Appendix C, “Server States” on page 169. You can automatically do the start attach by using the `_ASU_AUTOSTART` environment variable. See Chapter 4, “Creating the Environment Variable File” on page 21 for more information.

The system acknowledges the receipt of the request by displaying the message:

```
ASUV104I: Application Support server accepts Attach Facility Start request.
```

Note: Although the system acknowledges the receipt of the request to attach to a CICS or IMS subsystem, the panels do not indicate if the attachment was successful. To verify this, use the Display CICS/IMS Application Support Server Operational Data panel.

Limitations of Attaching the DCE AS IMS Server

In the case of DCE AS IMS, the Attach Facility only establishes an execution relationship between the Application Support server and VTAM. The actual communication with IMS is established only when the client makes an OPEN SESSION VTAM call to IMS. This means that even if the Application Support server is attached to an IMS region (and is in the operational state), it is not guaranteed that a session with IMS can be established. (A request to attach a nonexistent DCE AS IMS server to IMS will be successful.)

Stopping an Application Support Server Attachment

Selecting Option 2, Detach Application Support Server from Subsystem, in the primary options panel disables the Application Support Server specified in Server Name from accepting remote procedure calls on behalf of the CICS or IMS transactions that it supports. This facility provides for a controlled shutdown of the Application Support server to allow requests that are already in progress to complete. Selecting this option changes the state of the Application Support server from operational to quiescent. Server states are described in Appendix C, “Server States” on page 169.

The system acknowledges the receipt of the request by displaying the message:

ASUV105I: Application Support server accepts Attach Facility Stop request.

Note: Although the system acknowledges the receipt of the stop attachment request, the panels do not indicate if the attachment was successfully stopped. To verify this, use the Display CICS/IMS Application Support Server Operational Data panel.

Stopping and Restarting Attachment of DCE AS CICS Numerous Times

A virtual storage shortage situation may arise in the address space of the DCE AS CICS server if during the life of the server its attachment to the CICS subsystem is stopped and restarted numerous times. In this case, it is recommended that DCE AS CICS be recycled after stopping the attachment to the CICS system.

Displaying Interfaces and AS Server Operational Data

Selecting Option 3 from the primary options panel, Display Interfaces and Application Support Server Operational Data, displays a subpanel that allows you to perform the following:

- List all installed interfaces supported by the Application Support server
- Display information about a specific interface
- Display operational data about the Application Support server
- For DCE AS IMS only, display information about an IMS session.

Note: The fourth selection, Display Session Information, applies only to DCE AS IMS. Selecting this option in DCE AS CICS results in an error message.

Installing and Uninstalling Specific Interfaces

Selecting Option 4 from the primary options panel displays the Install / Uninstall Specific Interfaces panel (see Figure 26 on page 104) that you use to install or uninstall interfaces.

```

          Install / Uninstall Specific Interfaces
Server Name: /./AS/server_1

Enter 1 in the ACTION field to INSTALL an interface.
Enter 0 in the ACTION field to UNINSTALL an interface.

Then type the name of the interface in the INTERFACE NAME field.

ACTION    INTERFACE NAME  RESPONSE

  1        MMAXCSS
  -        _____
  -        _____
  -        _____
  -        _____

Command ==>
F1=Help   F2=Split   F3=Exit   F9=Swap   F12=Cancel

```

Figure 26. Install / Uninstall Specific Interfaces Panel

This panel has three fields: Action, Interface Name, and Response.

You have to fill up the Action and Interface Name fields with the appropriate Action codes interface names. The Response field indicates the status of the request made in the Action field.

Prerequisites to Interface Installation

Before an interface can be installed, the following must have been performed:

- Compile the server stub and link it with the DCE runtime routines to create the stub load module. Each server stub must be compiled and linked into a separate load module.
- Make sure that the stub library is accessible through the MVS LNKST concatenation or through a STEPLIB.
- The stub library must be APF-authorized if your server is running from an APF-authorized library. Your server runs from an APF-authorized library:
 - If you are using OTMA.
 - If you are using APPC and you set ASU_APF to YES.
 - If you are using CICS and transactional RPCs.

Optionally, you can create an RPC Profile to enable clients to search for the interface more efficiently.

For more information on compiling and linking interface stubs, see the *z/OS DCE Application Support Programming Guide*. Creating RPC Profiles is discussed in *z/OS DCE Administration Guide*.

Action Codes

The appropriate Action Code must be entered in the Action field. There are two Action Codes corresponding to either the install or uninstall request:

- 1 Installs an interface. This interface, as defined in the Interface Name field, is installed if the interface is found in the stub library and the interface has not already been installed. If the Application Support server is in the operational state, the interface is registered with the RPC Runtime. This advertises the availability of the interface to DCE clients.
- 0 Uninstalls an interface. The interface to be uninstalled must be in the installed state and the associated stub load module must be found and be valid. This is used to unadvertise the availability of an interface.

Interface Name

The interface name is the name of the stub load module that contains the interface definition, that is, the stub load module found in the STEPLIB.

Response Field

The Response field displays the status of the Install or Uninstall request.

Table 18 summarizes the possible responses and the meaning of each for an Install request.

<i>Table 18 (Page 1 of 2). Action Code 1: Responses and Meanings</i>	
Response	Meaning
Requested Action Successful	The interface name was found in the stub library and was found to be valid. It is exported to the directory. If the Application Support server is in the operational state, the interface is registered with the RPC Runtime. The interface name is added to the application profile.
An EPM error has occurred	An error occurred while exporting the interface with the end point map service.
An RPC runtime error has occurred	The interface cannot be registered with the RPC runtime.
A loader error has occurred	The Application Support server was unable to load the stub. The stub has likely been corrupted.
Unknown Response from App. Server	An unexpected response was encountered.
Interface already installed.	The interface name is already in the application profile. An interface with that name already exists.
DCE Directory error encountered	An error occurred while exporting the interface to the DCE directory.
Not installed. Too many interfaces.	500 interfaces have already been installed. No more interfaces can be installed. Check setting of <code>_EUV_LDR_MAX_XSTUBS</code> environment variable.
Interface is a duplicate of <i>interface-name</i>	Interface is not installed because it is a duplicate of another interface that is already installed. An interface with that UUID already exists.

<i>Table 18 (Page 2 of 2). Action Code 1: Responses and Meanings</i>	
Response	Meaning
Load module not found in STUB library	<p>Although the interface was in an uninstalled state, installation failed because of any of the following:</p> <ul style="list-style-type: none"> • The stub module was not found in the stub library. • The stub load module was not valid. • If your server is running from an APF-authorized library and your stub library is not APF-authorized. Your server runs from an APF-authorized library: <ul style="list-style-type: none"> – If you are using OTMA. – If you are using APPC and you set ASU_APF to YES. – If you are using CICS and transactional RPCs.
Interface is being uninstalled	A previous request to uninstall the interface is in progress, (that is, the interface is in the draining state). An interface can only be uninstalled after all its processes that are already in progress are completed.

Table 19 summarizes the possible responses and the meaning of each for an Uninstall request.

<i>Table 19. Action Code 0: Responses and Meanings</i>	
Response	Meaning
Requested Action Successful	The interface was found in the stub library. It is unexported from the directory. If the Application Support Server is in the operational state, the interface is deregistered with the RPC Runtime. The interface name is deleted from the application profile.
An EPM error has occurred	An error occurred while unexporting the interface with the end point mapper (EPM).
An RPC runtime error has occurred	The interface cannot be deregistered with the RPC runtime.
A loader error has occurred	The Application Support server was unable to unload the stub. The stub has likely been corrupted.
Unknown Response from App. Server	An unexpected response was encountered.
Interface not installed	The interface has previously been uninstalled successfully.
Installed - not in STUB library	The interface is installed but cannot be uninstalled because the load module is no longer in the stub library. Resolving this problem is discussed in the next section.

Installed Interfaces That Have Been Deleted

If the stub load module of an installed interface has been deleted from the stub library, that interface cannot be uninstalled. Also, if the Application Support server has been shutdown, any attempt to restart it fails. Uninstalling the interface involves deregistering the interface from the RPC Runtime. The UUID of the interface is required to unregister the interface and this UUID is contained in the stub load module. If the stub load module has been deleted, this UUID cannot be retrieved and thus, the interface cannot be uninstalled.

In this case, perform either of the following:

- Put the stub load module back in the library and retry the uninstall.

- Use the CDS control program to remove the Installed_IFs attribute of the interface. For example, to remove the Installed_IFs attribute of the interface **MNMAXCSS** for the Application Support server whose CDS object name is `./AS/server_1`:

```
cdscp> remove object ./AS/server_1 Installed_IFs = MNMAXCSS
```

The CDS control program is described in more detail in *z/OS DCE Command Reference*.

Note: Ensure that the **cds_attributes** file in the `/opt/dcelocal/etc` directory on the system you are on has the following two lines:

```
1.3.18.0.2.4.3 Installed_IFs      char
1.3.18.0.2.4.4 SubSystem_Info    char
```

If these lines do not appear in the file, add them to it.

Installing and Uninstalling Interfaces in Batch

You can also install or uninstall interfaces in batch by using the **ASUVIFM** utility. To be able to use this utility, the Application Support server to which the interfaces will be installed or uninstalled must be running. You must also perform a DCE Login as the Application Support server administrator.

Figure 27 is an example JCL that runs the ASUVIFM utility. In the PARM statement, specify 1 for install, or 0 for uninstall, followed by the CDS name of the Application Support server.

The example uses an inline data set that contains the interfaces that are going to be installed in the Application Support server `./AS/server_1`. This sample JCL is supplied as member ASUVIFM in data set `ASPREFIX.SASUSAMP`.

```
//JOBNAME JOB , 'Run ASUVIFM'
//*****
//*
//* Run ASUVIFM Bulk Installation Program
//*
//*****
//LOGIN EXEC PROC=DCELOGIN,
//      PARM='cell_admin -dce-'
//*
//ASUVIFM EXEC PGM=ASUVIFM, PARM='ENVAR('_EUV_HOME=/var/AS/server_1')
//      '1 ./AS/server_1'
//STEPLIB DD DSN=SMITH.DCE.LOAD, DISP=SHR
//SYSIN DD *
MNMAXCSS
intfc1
intfc2
/*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//CEEDUMP DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

Figure 27. Example JCL that Runs the ASUVIFM Utility

Working with Application Support Server Definitions

Selecting option 5 from the primary options panel displays the Work with Application Support Server Definitions panel, shown in Figure 28. Use this panel to perform the following tasks:

- Define an instance of the Application Support Server.
- Display data about an instance of the Application Support Server.
- Modify the definition of an instance of the Application Support Server.
- Delete an instance of the Application Support Server.

```
Work with Application Support Server Definitions
Type an Application Support Server name. Select an Option. Then press Enter.
Server Name: /./AS/server_1_____
_____
_____
1 1. Create an Application Support Server Definition
    Select a Subsystem for the Create option.
        1 1. CICS
          2. IMS with ISC
          3. IMS with APPC
          4. IMS with OTMA
    2. Display an Application Support Server Definition
    3. Modify an Application Support Server Definition
    4. Delete an Application Support Server Definition

Command ==>>
F1=Help  F2=Split  F3=Exit  F9=Swap  F12=Cancel
```

Figure 28. Work with Application Support Server Definitions Panel

Note: Options 2 (Display an Application Support Server Definition) and 3 (Modify an Application Support Server Definition) may not appear. This happens if you have specified the name of an existing APPC or OTMA server on the main panel (see Figure 23 on page 100). To create a new definition, you can type over the server name on this panel.

Defining an Application Support Server

When option 1 of this panel, Create an Application Support Server Definition, is to be selected, you must specify the subsystem to which the Application Support server is going to be attached. Enter 1 for CICS, or 2 for IMS on the space provided for selecting the appropriate subsystem.

Defining an Application Support Server for CICS: In DCE AS CICS, the Create a CICS Application Support Server Definition panel shown in Figure 29 is displayed.

```

Create a CICS Application Support Server Definition

Server Name: /./AS/server_1

CICS Generic Application ID: OETCICP4
Server Generic Application ID: CICSAS1

Command ==>
F1=Help   F2=Split  F3=Exit   F9=Swap   F12=Cancel

```

Figure 29. Create a CICS Application Support Server Definition Panel

The following explains each field and the possible values that can be entered.

Server Name	This is the server name specified in the primary options panel and is automatically displayed by this panel.
CICS Generic Application ID	The Application Support Server uses this to locate the CICS subsystem to which the Application Support Server is to attach when the Attachment Facility is started. This has the same value as the APPLID assigned to a specific instance of CICS. This value is defined in the System Initialization Table (SIT) as an APPLID= <i>keyword</i> statement.
Server Generic Application ID	The Application Support Server uses this to locate the CONNECTION definition in the CICS subsystem that corresponds to the specified CICS Generic Application ID. The CONNECTION definition specifies the number and characteristics of the sessions between the Application Support Server and the target CICS subsystem. The Server Generic Application ID must be specified as the NETNAME assigned to a CONNECTION definition.

Defining an Application Support Server for ISC: In DCE AS IMS with ISC, the Create an IMS Application Support Server Definition panel shown in Figure 30 on page 110 is displayed.

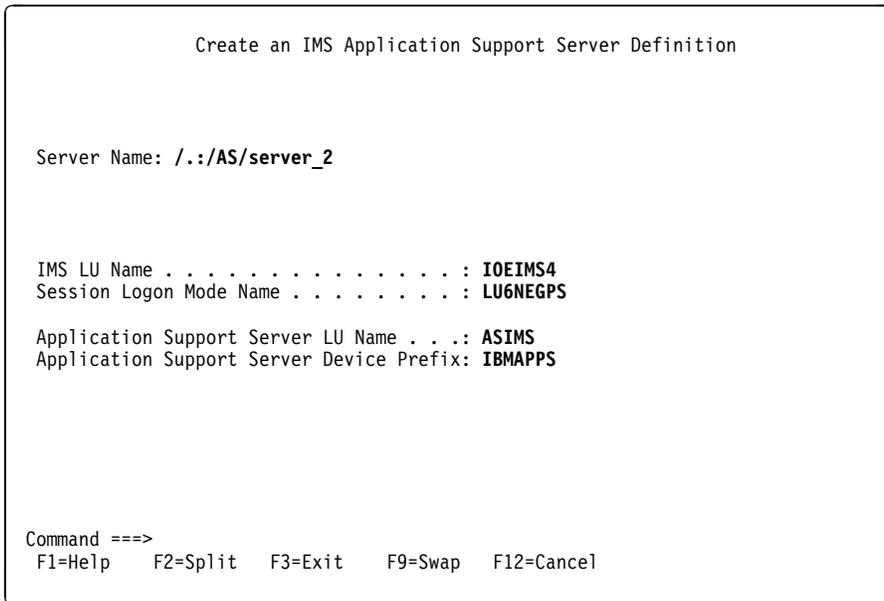


Figure 30. Create an IMS Application Support Server Definition Panel

Note: This panel appears only for AS IMS with ISC, not for AS IMS with APPC or OTMA.

The following explains each field and the possible values that can be entered.

Server Name	This is the server name specified in the primary options panel and is automatically displayed by this panel.
IMS LU Name	This specifies the LU name of the target IMS subsystem that will communicate with the Application Support server.
Session Logon Mode Name	This specifies the name of the entry in the VTAM logon mode table.
Application Support Server LU Name	This specifies the LU name assigned to the instance of the Application Support Server.
Application Support Server Device Prefix	This specifies a prefix that is used in generating the name of the session endpoint. The connection between the Application Support server and the target IMS subsystem is defined as LUTYPE6 (ISC) with parallel sessions to IMS. IMS requires that each of the source endpoints be identified as a logical device. The list of logical devices are defined to the IMS subsystem. The generated names of the session endpoint are used in defining the IMS logical devices or subpools. The Application Support server assumes the logical device names consist of a constant device prefix and a monotonically increasing suffix. The suffix is a hexadecimal number beginning from 0 up to the maximum number of sessions. A maximum of 7 characters may be used for the device prefix. For example, if the device prefix is IBMAPPS, session endpoint names IBMAPPS0, IBMAPPS1, and so forth, are generated when a session is required by the Application Support server.

Displaying Data About an Application Support Server

Selecting option 2 from the Work with Application Support Server Definitions panel displays the Display a CICS Application Support Server Definition or the Display an IMS Application Support Server Definition panel, depending on whether you are using CICS or IMS.

Modifying Data about an Application Support Server

(This is for ISC or CICS servers only.) Selecting option 3 of the Work with Application Support Server Definitions panel displays the Modify a CICS Application Support Server Definition panel or the Modify an IMS Application Support Server Definition panel, depending on whether you are using CICS or IMS. Use these panels to modify subsystem information.

Deleting the Definition of an Application Support Server

Selecting option 4 of the Work with Application Support Server Definitions panel displays the Delete a CICS Application Support Server Definition panel or the Delete an IMS Application Support Server Definition panel, depending on whether you are using CICS or IMS. Use these panels to delete a particular instance of an Application Support server.

Removing Stale Cache Entries: If you change Application Support server characteristics, the CDS information on the Application Support server in the CDS cache becomes stale. Any DCE client call to this Application Support server fails if the client retrieves the binding information from the CDS cache.

This situation can be resolved by either:

- Having the client log in to DCE again. Doing so creates a new credentials cache file, the opaque form of the user's login context. Because the key to access the client cache is the new credentials cache file, the user cannot access the previous (stale) cache entries.
- Using the CDS control program to set the **cdscp confidence** to either **medium** or **high**. Then, use the **show object** subcommand of the CDS control program on the server entry of the Application Support server. Because of the **cdscp confidence** setting, the CDS Clerk obtains information about the Application Support server directly from the CDS server and refreshes the CDS cache.

For example, for Application Support server `./AS/server_1`, the client can run the following CDS control program commands:

```
cdscp> set cdscp confidence = medium
cdscp> show object ./AS/server_1
```

Redefining Access Privileges to the Application Support Server Namespace

Entry: When the definition of the Application Support server is deleted, the ACL information of the Application Support server's namespace entry is also deleted. If this server is to be redefined to CDS and restarted, the definition of the ACL permissions to the Application Support server's namespace entry (as described in Chapter 11, "Configuring the CDS Namespace Entries" on page 63) must be performed again.

Logging In to z/OS DCE

Selecting option 6 from the primary options panel allows you to login to DCE. Normally, you login to DCE using the DCELOGIN command from TSO or `dce_login` from the shell. However, you may want to use this option if:

- You started the Application Support Server Administration Panels without logging in to DCE. If you have not logged in to DCE, the DCE Security Service has not authenticated you, and you cannot perform most of the administrative functions these panels provide. Using this option, you can login to DCE without exiting from the administration panels.
- You want to perform the administrative tasks under a different user ID than the one that you are currently logged on. Using this option, you can change the DCE userid that you are logged on without exiting from the administration panels.

When you select this option, the DCE Login Panel shown in Figure 31 is displayed.

Perform DCE Login

Type an Administrator Principal Name and Password. Then press Enter.

Administrator Principal Name : _____

Administrator Principal Password:

Command ==>
F1=Help F2=Split F3=Exit F9=Swap F12=Cancel

Figure 31. Perform DCE Login Panel

You may enter up to 256 characters in both the DCE Userid and DCE Password fields. The DCE Password that you enter is not displayed on the screen.

The following message is displayed if you logged in successfully to DCE:

```
ASUV00234I Logon is successful
```

If the login is unsuccessful, an error message is displayed. The *z/OS DCE Messages and Codes* explains these messages.

Reloading the Identity Mapping File

Selecting option 7 from the primary options panel lets you reload the identity mapping file. If you change the Identity Mapping file after it is used, the RELOAD function can make the changes take effect. When you select Reload Identity Mapping File from the primary options panel, the Identity Mapping file in use is replaced with the updated version containing your changes. See “Modifying and Deleting Identity Mapping Entries” on page 60 for more information.

Chapter 18. Administering the AS Server Using MVS System Commands

You can perform the following Application Support server administration tasks from the MVS operator console:

- Starting attachment of an Application Support server
- Stopping attachment of an Application Support server
- Reloading the identity mapping file
- Displaying the server's operational status
- Displaying the status of one or all interfaces
- Installing or uninstalling an interface
- Stopping the Application Support server

See Table 17 on page 97 for a table referencing information about alternative ways to perform administrative tasks.

Notes:

1. You do **not** have to login to DCE to be able to perform these tasks from the console.
2. If you enter an MVS MODIFY system command, the response goes to the MVS console. However, if there are any errors, these messages are directed as specified in the `_EUV_SVC_MSG_LOGGING` environment variable.

Starting Attachment of an Application Support Server

The Application Support server must be **attached** to the CICS or IMS subsystem for it to be in a state where it can receive remote procedure calls to the CICS or IMS transactions that it supports. “Attaching the Application Support Server” on page 102 describes how to use the administration panels to attach an Application Support server to a CICS or IMS subsystem.

The Application Support server can also be attached to CICS or IMS using system commands. An Application Support server attachment is started using the **MODIFY (F)** system command. To start the attachment, enter the following command:

```
MODIFY jobname,STARTATT
```

jobname

is the MVS name of the Application Support server batch job or started task.

This command is equivalent to selecting option 1 of the primary options administration panel, Attach Application Support Server to Subsystem.

This command can only be run if the Application Support server has been started and initialized.

The system acknowledges the receipt of the request by displaying the message:

```
ASUV0104I Application Support server accepts Attach Facility Start request.
```

Note that although the system acknowledges the receipt of the attach request, this does not necessarily indicate that attachment is successful. You can verify if the attachment was successful by using the Display Interfaces and Server Operational Data administration panel.

If the attachment was not successful, an error message (messages are listed in Appendix G, “Application Support Server Messages and Codes” on page 193) is displayed on the console.

Stopping Attachment of an Application Support Server

The attachment of the Application Support server to CICS or IMS may have to be stopped for a variety of reasons, for example, if the Application Support server is to be moved to another CICS or IMS subsystem.

You can also stop the attachment of the Application Support server from CICS or IMS using the **MODIFY** system command. To stop the attachment, enter the following command:

MODIFY *jobname*,**STOPATT**

jobname

is the MVS name of the Application Support server batch job or started task.

This command is equivalent to selecting option 2 of the primary options administration panel, Detach Application Support Server from Subsystem.

This command can only be run if the Application Support server has been attached to CICS or IMS.

The system acknowledges the receipt of the request by displaying the message:

ASUV0105I Application Support server accepts Attach Facility Stop request.

Note that although the system acknowledges the receipt of the stop attachment request, this does not necessarily indicate that it is successful. You can verify if the request was successful by using the Display Interfaces and Server Operational Data administration panel.

If the request for stop attachment was not successful, an error message (messages are listed in Appendix G, “Application Support Server Messages and Codes” on page 193) is displayed on the console.

Reloading the Identity Mapping File with the MODIFY Command

If you make changes to the Identity Mapping file, you need to call the **RELOAD** function to activate the changes you made. You can use the **MODIFY** system command to perform the RELOAD function. To reload the Identity Mapping file, enter the following command:

MODIFY *jobname*,**RELOAD**

jobname

is the MVS name of the Application Support server batch job or started task.

This command can only be run if the Application Support server has been started and initialized.

Note to RACF Users: If `_ASU_EX_SEC` is set to YES, the **RELOAD** function ends with a message indicating that **RELOAD** is not supported for RACF systems. See “Mapping DCE Principal Names to MVS User IDs with RACF” on page 60 for more information on RACF support.

Displaying the Server's Operational Status

You may want to display the operational status of the Application Support server. You can do this using the MODIFY system command:

MODIFY *jobname*,DISPSRVR

jobname

is the MVS name of the Application Support server batch job or started task.

The following figures illustrate the format of output and then provide samples of output for the various Application Support server types (CICS, IMS ISC, IMS APPC, and IMS OTMA). Figure 32 shows the format for output:

```
+ASUV00815I cds_name,
server_type,server_state,server_start_time,server_level,
attach_start/stop_time,userid,message_num,num_interfaces,num_threads,
server_specific_information
```

Figure 32. DISPSRVR Output Format

cds_name	is the name of the AS server. Up to 55 characters of the name are displayed.
server_type	is the type of the server. It can be CICS, IMS ISC, IMS APPC, or IMS OTMA.
server_state	is a number that represents the current state of the AS server. <ul style="list-style-type: none">1 - initializing2 - quiescent3 - starting4 - operational5 - stopping6 - terminating See Appendix C, "Server States" on page 169 for descriptions of these states. Note: If the Application Support server is not yet attached to the CICS or IMS subsystem and the server_state is 1 or 2, the output could look like the OTMA sample in Figure 36 on page 119.
server_start_time	is the start time of the AS server. This is in universal time coordinated (UTC) rather than local time.
server_level	is the date and time, in <i>yyyymmddhh.mm</i> -format, when IBM created the linkage editor input for the creation of the server load module. If you are changing the Customization Component, you can control the displayed information by using the BLDDATE function. (For information about the BLDDATE function, see the <i>z/OS DCE Application Support Programming Guide</i> .) The time is in local time rather than UTC.
attach_start/stop_time	is the time the attachment facility was started or stopped. This is in UTC rather than local time. This message variable does not appear if the attachment facility was never started or stopped.

userid	is the MVS user ID of the administrator who started or stopped the attachment facility. This is the user ID associated with the DCE principal name that issued the dce_login command, not the user ID the administrator used to log on to TSO/E. If this was done from the operator console, or if the attachment facility was automatically started because the value of the environment variable _ASU_AUTOSTART is YES, the user ID in this message is MVSOP. This message variable does not appear if the attachment facility was never started.
message_num	is the number of the message that gives the reason why the attachment facility is in its current state. See Appendix G, "Application Support Server Messages and Codes" on page 193 for descriptions of these messages.
num_interfaces	is the number of installed interfaces.
num_threads	is the number of executor threads. This is the number of threads that are available to process RPCs to the AS server.
server_specific_information	is information that is unique to the type of server that is running. <ul style="list-style-type: none"> For a CICS server, this consists of cics_applid,server_applid. <ul style="list-style-type: none"> cics_applid is the CICS APPLID. server_applid is the AS server generic APPLID. For an IMS ISC server, this consists of ims_luname,logmode,as_luname,device_prefix. <ul style="list-style-type: none"> ims_luname is the IMS logical unit name. logmode is the logon mode table entry name. as_luname is the AS server logical unit name. device_prefix is the device prefix. For an IMS APPC or OTMA server, there is no unique information.

Figure 33 shows sample output for an AS CICS server.

```

JOB00142 00000090 +ASUV00815I /./AS/ASIMS51, 439
      439 00000090 CICS,4,Fri Aug 2 18:55:06 1996,1996080214.45,
      439 00000090 Fri Aug 2 18:58:19 1996,MVSOP,ASUV00129,2,20,
      439 00000090 OETCICP4,CICSAS1

```

Figure 33. DISPSRVR Sample Output for AS CICS Server

Figure 34 shows sample output for an AS IMS ISC server.

```

JOB00142 00000090 +ASUV00815I /./AS/ASIMS51, 439
      439 00000090 IMS ISC,4,Fri Aug 2 18:55:06 1996,1996080214.45,
      439 00000090 Fri Aug 2 18:58:19 1996,MVSOP,ASUV00129,2,1,
      439 00000090 IOEIMS4,LU6NEGPS,ASIMS,IBMAS

```

Figure 34. DISPSRVR Sample Output for AS IMS ISC Server

Figure 35 on page 119 shows sample output for the AS IMS APPC server.

```
JOB00142 00000090 +ASUV00815I ./:/AS/ASIMS51, 439
439 00000090 IMS APPC,4,Fri Aug 2 18:55:06 1996,1996080214.45,
439 00000090 Fri Aug 2 18:58:19 1996,MVSOP,ASUV00129,0,64
```

Figure 35. DISPSRVR Sample Output for AS IMS APPC Server

Figure 36 shows sample output for the AS IMS OTMA server. (For this example, the server state is 2.)

```
JOB00128 00000090 ASUV00815I ./:/AS/eve/DCEIMS/otma_1, 157
157 00000090 IMS OTMA,2,Tue Feb 18 18:30:49 1997,1997020515.21,
157 00000090 ,,ASUV00128,9,10
```

Figure 36. DISPSRVR Sample Output for AS IMS OTMA Server

Displaying the Status of Interfaces

You may want to display the status of one or all interfaces. You can do this using the MODIFY system command:

```
MODIFY jobname,DISPIF [interfacename]
```

jobname is the MVS name of the Application Support server batch job or started task.

interfacename is the 1- to 8-character name of the interface to display. This is the name of a partitioned data set (PDS) or partitioned data set extended (PDSE) member that exists in the stub library that the STEPLIB DD statement defines for the Application Support server job. If you omit the name of the interface, this command displays the names of all installed interfaces.

Figure 37 shows the format of output when you specify an interface name:

```
ASUV00816I ifname,ifstate,major_num.minor_num,iftype,ifnum_ops
```

Figure 37. Format of Output for Displaying the Status of Interfaces

ifname is the name of the interface.

ifstate is a number that represents the current state of the interface:

0 interface is not installed.

1 interface is installed.

2 interface is installed but cannot be found in stub library.

3 interface is draining.

major_num is the major version number.

minor_num is the minor version number.

iftype is a number that represents the type of interface.

1 CICS

2 IMS

An asterisk (*) immediately following the interface type number indicates that the interface is transactional.

ifnum_ops is the number of operations contained in the interface definition.

Note: Only the first two message variables appear if the interface state is 0, 2, or 3.

The following shows an example of output (assuming that the user specified an *interface*):

```
ASUV00816I TRPCMIN,1,1.0,2*,3
```

Figure 38 shows an example of output (assuming that the user did not specify an *interface*):

```
ASUV00817I INTRFAC1 INTRFAC2 INTRFAC3 INTRFAC4 INTRFAC5 INTRFAC6
ASUV00817I INTRFAC7 INTRFAC8 INTRFAC9 INTRFACA INTRFACB INTRFACC
:
ASUV00817I INTRFACn INTRFACn INTRFACn INTRFACn INTRFACn INTRFACn
```

Figure 38. DISPIF Sample Output for Multiple Interfaces

The list of interfaces is sorted alphabetically before the first ASUV00817I message is issued. Up to six interface names are displayed for each message, with eight positions allotted for each name (for columnar output).

Installing and Uninstalling Interfaces

You can use the MODIFY system command to install or uninstall a specific interface: The format for installing is:

```
MODIFY jobname,INSTIF interfacename
```

The format for uninstalling is:

```
MODIFY jobname,UNINSTIF interfacename
```

jobname is the MVS name of the Application Support server batch job or started task.

interfacename is the name of a PDS or PDSE member in the stub library that the STEPLIB DD statement defines for the Application Support server job.

For INSTIF and UNINSTIF, if the requested operation is successful, message ASUV00902I is displayed on the console. In all other cases, a message in the range of ASUV00903I through ASUV00914A is issued.

Stopping the Application Support Server

To stop the Application Support server, use the **STOP (P)** system command:

```
STOP jobname
```

jobname is the name of the Application Support server batch job or started task.

To run this command successfully, the Application Support server must be in either the operational or the quiescent state.

The following message is displayed if the Application Support server is successfully stopped:

```
ASUV0812I Application Support server stops.
```

If the Application Support server is not stopped successfully, an error message (messages are listed in Appendix G, “Application Support Server Messages and Codes” on page 193) is displayed on the console.

Chapter 19. Administering the AS Server Using a Client Program

A customer-written client program can use APIs to perform Application Support server administrative tasks:

vensrv_mgmt_display_if	Displays the status of an interface.
vensrv_mgmt_display_ifs	Queries the interfaces installed on an Application Support server.
vensrv_mgmt_display_server	Queries the server's data or operational status.
vensrv_mgmt_if_mgmt	Installs or uninstalls an interface.
vensrv_mgmt_reload_int	Reloads the identity mapping file.
vensrv_mgmt_start_attachment	Starts attachment of the Application Support server.
vensrv_mgmt_stop_attachment	Stops attachment of the Application Support server.

You must `dce_login` as a principal who has the appropriate permissions to access the Application Support server administration program. See Chapter 12, "Giving Access to the Administration Program" on page 67 for details about access. See Appendix F, "Administrative APIs Example Client Program" on page 181 for an example of a client program.

You can also perform the preceding tasks using administration panels or MVS system commands. See Table 17 on page 97 for a list of different ways to perform administrative tasks.

What You Need to Build the Client

To build the client, you need the following files from `/usr/lpp/dceas/share/include/admin`:

- `asvsmif.idl`
- `asvsmif.acf`
- `asvsmdf.idl`

The definitions for the return codes are in the shipped IDL file `asvsmdf.idl`.

No system responses are issued at the console when the client code calls any of the administrative APIs.

For information about writing and building a client application, see *z/OS DCE Application Development Guide: Introduction and Style*.

vensrv_mgmt_display_if

Description

This API displays the status of a specified interface. You can write the client program to show the data associated with the interface by interpreting the interface data that the call returns.

Format

```
[idempotent]
error_status_t vensrv_mgmt_display_if
(
    [in] handle_t h,
    [in] mvs_name_t *if_name,
    [out] xif_data_t *ifdata,
    [out] vmgmt_rc_t *mrc
);
```

Parameters

h Is the binding handle.

if_name
Is the 8-byte name of the interface to display.

ifdata
Is a pointer to a structure as follows and is the data associated with an interface:

ifdata.major_version
Is the major version number.

ifdata.minor_version
Is the minor version number.

ifdata.num_ops
Is the number of operations defined in the interface.

ifdata.if_type
Is a number representing the interface type.

mrc
Is a pointer to the following structure:

mrc.rc
Is the return code, which can be one of the following:

```
vmgmt_rc_ok
vmgmt_if_not_installed
vmgmt_if_inst_nt_found
vmgmt_if_uninst_in_progress
vmgmt_authentication_failure
```

mrc.state
Is the current server state.

mrc.state_reason
Is the current reason code.

mrc.error_corr

Is the error correlator.

Return Values

error_status_t

Is the returned status:

Return Value	Meaning
error_status_ok	Success
All other values	See the shipped IDL file asuvmf.idl for details.

vensrv_mgmt_display_ifs

Description

This API queries the current interfaces installed on an Application Support server.

Note: You can write the client program to show the names of the interfaces installed by interpreting the interface structure that is returned.

Format

```
[idempotent]
error_status_t vensrv_mgmt_display_ifs
(
    [in] handle_t h,
    [out] xif_vector_t *ifs,
    [out] vmgmt_rc_t *mrc
);
```

Parameters

h Is the binding handle.

ifs Is a pointer to a structure as follows:

ifs.number_of_ifs

Is the total number of interfaces installed.

ifs.name[]

Is an array of interface names. The number of elements in the array is *ifs.number_of_ifs*.

mrc

Is a pointer to the following structure:

mrc.rc

This is the return code, which can be one of the following:

vmgmt_rc_ok

vmgmt_if_not_installed

vmgmt_if_inst_nt_found

vmgmt_if_uninst_in_progress

vmgmt_authentication_failure

mrc.state

Is the current server state.

mrc.state_reason

Is the current reason code.

mrc.error_corr

Is the error correlator.

Return Values

error_status_t

Is the returned status:

Return Value	Meaning
error_status_ok	Success
All other values	See the shipped IDL file asvmdf.idl for details.

vensrv_mgmt_display_server

Description

This API displays the operational status of the Application Support server or queries current server data.

Note: You can write the client program to show the data and status of the server by interpreting the `server_data` that the call returns.

Format

```
[idempotent]
error_status_t vensrv_mgmt_display_server
(
    [in] handle_t h,
    [out] vserver_data_t *server_data,
    [out] vmgmt_rc_t *mrc
);
```

Parameters

h Is the binding handle.

server_data

Is a pointer to the following structure:

server_data.number_executors

Is the number of executor threads created.

server_data.number_xifs_installed

Is the number of interfaces installed.

server_data.server_princ_name

Is the server principal name.

server_data.who_started_stopped_attachment

Is the name of the issuer of the start or stop attachment.

server_data.mvs_id_start_stop_attachment

Is the MVS user ID of the last one to start or stop the attachment.

server_data.server_start_time

Is the time the attachment started.

server_data.started_stopped_time

Is the time the attachment started or stopped.

server_data.subsystem_info[]

Is 33 characters of subsystem information.

server_data.genned

Is the date and time the server was generated.

server_data.server_cds_name

Is the server CDS name.

mrc

Is a pointer to the following structure:

mrc.rc

Is a return code, which can be one of the following:

vmgmt_rc_ok
vmgmt_call_rejected
vmgmt_authentication_failure

mrc.state

Is the current server state.

mrc.state_reason

Is the current reason code.

mrc.error_corr

Is the error correlator.

Return Values

error_status_t

Is the returned status:

Return Value	Meaning
error_status_ok	Success
All other values	See the shipped IDL file asuvmdf.idl for details.

vensrv_mgmt_if_mgmt

Description

This API installs or uninstalls one or more interfaces.

Format

```
[idempotent]
error_status_t vensrv_mgmt_if_mgmt
(
    [in] handle_t h,
    [in] xif_mgmt_vector_t *ifs,
    [out] xif_vector_t *ifs_duplicate,
    [out] xif_vector_status_t *update_rc,
    [out] vmgmt_rc_t *mrc
);
```

Parameters

h Is the binding handle.

ifs Is a pointer to a structure as follows:

ifs.action

Is one of the following:

vmgmt_if_install
vmgmt_if_uninstall

ifs.name[]

Is an array of interface names that you wish to install or uninstall.

ifs_duplicate

Is a pointer to a structure of the interfaces that are duplicated. Look at *ifs_duplicate* if the return code status of *update_rc* is *vmgmt_if_duplicate_if*.

ifs_duplicate.number_of_xifs

Is the number of duplicate interfaces returned.

ifs_duplicate.name[]

Is an array of interface names. The number of elements in the array is *ifs_duplicate.number_of_xifs*.

update_rc

Is a pointer to the following structure:

update_rc.number_of_xifs

Is the number of interfaces installed.

update_rc.status[]

Is an array of status codes. The status codes can be one of the following if the function is to install an interface.

```

vmgmt_if_ok
vmgmt_if_too_many
vmgmt_if_not_found
vmgmt_if_already_inst
vmgmt_if_dir_error
vmgmt_if_epm_error
vmgmt_if_rt_error
vmgmt_if_duplicate_if
vmgmt_if_uninst_in_progress
vmgmt_if_loader_error

```

It can be one of the following if the function is to *uninstall* an interface:

```

vmgmt_if_ok
vmgmt_if_not_installed
vmgmt_if_dir_error
vmgmt_if_epm_error
vmgmt_if_rt_error
vmgmt_if_loader_error

```

It can be the following if the function is undefined:

```

vmgmt_call_rejected

```

mrc

Is a pointer to the following structure:

mrc.rc

Is the return code, which can be one of the following:

```

vmgmt_if_ok
vmgmt_call_rejected
vmgmt_authentication_failure

```

mrc.state

Is the current server state.

mrc.state_reason

Is the current reason code.

mrc.error_corr

Is the error correlator.

Return Values

error_status_t

Is the returned status:

Return Value	Meaning
error_status_ok	Success
All other values	See the shipped IDL file asuvmdf.idl for details.

vensrv_mgmt_reload_imt

Description

This API calls the RELOAD function to activate changes you have made to the Identity Mapping file.

Format

```
[idempotent]
error_status_t vensrv_mgmt_reload_imt
(
    [in] handle_t    h,
    [out] vmgmt_rc_t *mrc
);
```

Parameters

h Is the binding handle.

mrc

Is a pointer to the following structure:

mrc.rc

Is the return code, which can be one of the following:

```
vmgmt_rc_ok
vmgmt_call_rejected
vmgmt_authentication_failure
vmgmt_cant_access_idmap_file
vmgmt_cant_build_hash_table
```

mrc.state

Is the current server state.

mrc.state_reason

Is the current reason code.

mrc.error_corr

Is the error correlator.

Return Values

error_status_t

Is the returned status:

Return Value	Meaning
error_status_ok	Success
All other values	See the shipped IDL file asuvmdf.idl for details.

vensrv_mgmt_start_attachment

Description

The Application Support server must be **attached** to the CICS or IMS subsystem for it to be in a state where it can receive remote procedure calls to the CICS or IMS transactions that it supports. This API starts the attachment.

Format

```
[idempotent]
error_status_t vensrv_mgmt_start_attachment
(
    [in] handle_t h,
    [out] vmgmt_rc_t *mrc
);
```

Parameters

h Is the binding handle.

mrc

Is a pointer to a structure:

mrc.rc

Is the return code, which can be one of the following:

```
vmgmt_rc_ok
vmgmt_call_rejected
vmgmt_attachment_error
vmgmt_authentication_failure
```

mrc.state

Is the current server state.

mrc.state_reason

Is the current reason code

mrc.error_corr

Is the error correlator.

Return Values

error_status_t

Is the returned status:

Return Value	Meaning
error_status_ok	Success
All other values	See the shipped IDL file asuvmdf.idl for details.

vensrv_mgmt_stop_attachment

Description

The attachment of the Application Support server to CICS or IMS may have to be stopped for a variety of reasons, for example, if the Application Support server is to be moved to another CICS or IMS subsystem. This API stops the attachment.

Format

[idempotent]

```
error_status_t vensrv_mgmt_stop_attachment
(
    [in] handle_t h,
    [out] vmgmt_rc_t *mrc
);
```

Parameters

h Is the binding handle.

mrc

Is a pointer to a structure:

mrc.rc

Is the return code, which can be one of the following:

```
vmgmt_rc_ok
vmgmt_call_rejected
vmgmt_attachment_error
vmgmt_authentication_failure
```

mrc.state

Is the current server state.

mrc.state_reason

Is the current reason code.

mrc.error_corr

Is the error correlator.

Return Values

error_status_t

Is the returned status:

Return Value	Meaning
error_status_ok	Success
All other values	See the shipped IDL file asuvmf.idl for details.

Chapter 20. Using the tkadmin Command with the AS Server

The AS CICS or AS IMS OTMA server can be an Encina-based server. If you are using transactional RPCs, you can use Encina **tkadmin** commands from a non-z/OS platform to determine the state of the server and Encina transactions.

You can direct any Encina **tkadmin** command—except the **tkadmin stop server** command, which is specifically disallowed—at the AS CICS or IMS OTMA server. The same ACL that controls use of the Application Support administration program controls use of the **tkadmin** commands. Although you can enter any **tkadmin** command to the Application Support server, some commands have no meaning. For example, commands for setting up and managing a server's log file, managing volumes for a server's data, or creating and deleting logical volumes each return an error message to the caller of the command or produce incorrect or no output.

Encina **tkadmin** commands are of two types:

- Those commands that do not change the target server's environment, such as variations of **list** and **query**. DCE principals that are allowed to issue these commands must have **read (r)** permission in the Application Support administration program ACL. (This permission correlates with Encina **query (q)** authority.)
- Those that could change the target server's environment, such as variations of **set** and **trace**. DCE principals that are allowed to issue these commands must have **execute (x)** permission in the Application Support administration program ACL. (This permission correlates with Encina **administer (a)** authority.)

You must enter **tkadmin** commands from a non-z/OS platform with Encina installed (such as AIX®). This is because the z/OS Encina Toolkit Executive does not supply an OS/390 or z/OS version of the **tkadmin** command. See the Administrator's Guide for the Encina Toolkit for your system. for a description of the **tkadmin** command. See Chapter 12, "Giving Access to the Administration Program" on page 67 for more information about defining the Application Support administration program ACL.

Chapter 21. Managing Transactions Called with Transactional RPCs

AS transactional RPC support for IMS or CICS transactions lets AS IMS OTMA servers or AS CICS servers participate in Encina⁵ distributed transaction processing applications. An Encina *transaction* is a set of operations that must be done as a single unit for consistent transformation of data. In a typical Encina transaction processing scenario, a client application begins a transaction, accesses one or more remote servers that manage recoverable data, and then ends the transaction.

AS transactional support lets the client application include IMS or CICS transactions within the scope of an Encina transaction. Data integrity is guaranteed across all servers (or participants) involved. If any participant, including AS IMS or AS CICS, is unable to perform the required operations, all of the work is undone (aborted or rolled back). Updates to recoverable data are committed only if all participants can successfully perform their work. Once committed, updates are not lost in subsequent system failures.

The z/OS Encina Toolkit Executive enhances DCE with transactional semantics. It implements a two-phase commit protocol. This is a set of actions that ensures an application program makes *all* the changes of a transaction to a collection of resources or makes *no* changes. The z/OS Encina Toolkit Executive is a subset of Transarc's Encina.

Although the z/OS Encina Toolkit Executive supports only ephemeral clients (those with no facility for logging results or recovery), Application Support provides recovery by using native z/OS RRMS and IMS or CICS facilities in conjunction with z/OS Encina Toolkit Executive services.

The following figure illustrates potential components in resolving the outcome of an Encina transaction using AS IMS or AS CICS. Note that the figure does **not** depict any application data flow.

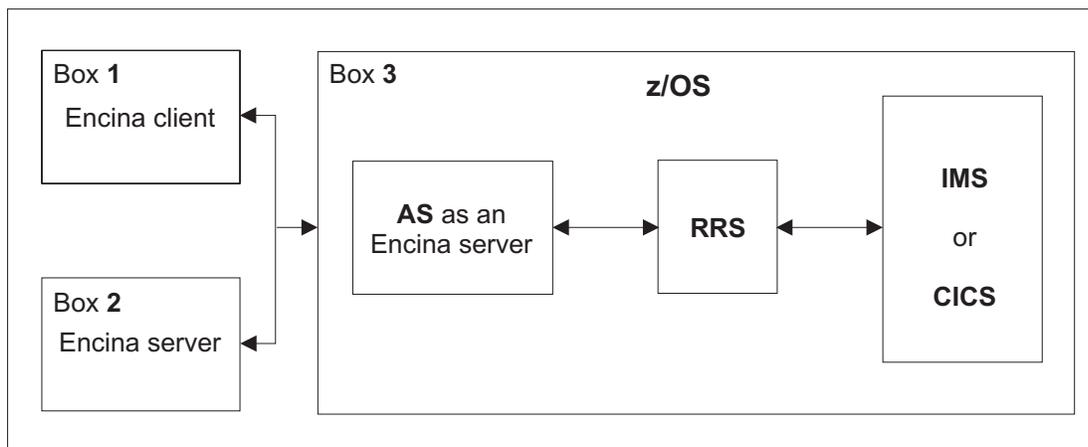


Figure 39. Components Resolving an Encina Transaction

z/OS DCE Application Support servers are specialized Encina servers. The Encina protocol controls the two-phase commit for the entire distributed transaction. RRS coordinates the local outcome for an Application Support server and the CICS or IMS subsystem.

⁵ Encina is a family of products that provides a range of services for developing and deploying large-scale client-server and open, distributed, transaction-processing systems. Layered on top of OSF DCE, Encina uniquely extends DCE with services, such as transactions, that are necessary when building business-critical systems.

For this chapter, you need to understand the following terms:

Application Support server (AS server)

For this chapter, this means an AS CICS or AS IMS with OTMA server.

recoverable Encina server

An Encina server with facilities for logging information and recovery.

ephemeral application

An Encina application with no facilities for logging information or recovery.

participant

An application that initiates a transaction or receives a request on behalf of the transaction.

coordinator

The application that polls all participants to determine if a transaction commits or aborts.

server distributed synchronization point resource manager (SDSRM)

The term RRS uses to describe the role of a resource using a client/server protocol. Application Support takes on this role when it coordinates Units of Recovery. See *z/OS MVS Programming: Resource Recovery* for more information.

two-phase commit protocol

A protocol to guarantee that either all updates are committed or all updates are backed out. Both Encina and RRS implement this protocol. See “The Two-Phase Commit Protocol” on page 139 for details.

Encina transaction

A set of operations that must be done as a single unit to ensure data consistency.

unit of recovery

The work done by and for a context between one point of consistency and another.

When a client uses DCE RPC to call an IMS or CICS transaction through an AS server, IMS or CICS commits or backs out any update done before returning to the client. Transactional RPCs differ from nontransactional RPCs in who controls the transaction outcome (that is, commit or abort). With nontransactional RPCs, IMS or CICS makes the decision to commit or abort. With transactional RPCs, the client may call other servers and eventually commits or aborts all updates including updates to IMS or CICS. Each server must be a recoverable Encina server to participate in the coordination. When the commit or abort is issued, Encina selects one of the servers as the coordinator and the other servers are participants. (Because CICS and IMS are not direct participants in the global Encina transaction, Encina never selects IMS or CICS to coordinate the transaction.)

The two-phase commit flows are then sent to each participant and coordinator resource. These two-phase commit flows happen automatically and, at the end of the two phases, either all updates have been committed or all have rolled back.

There may be times, however, when problems occur during the two-phase commit processing. These may be caused by network outages, outage of the coordinator server, outage of the AS server, an outage of RRS where the AS server is running, or an outage of IMS or CICS itself. The easiest and safest recovery from these conditions is to fix the failure, for example, bring up the failed server or servers, or fix

the network outage. Encina then automatically completes the two-phase commit. However, there may be times when the administrator may need to use manual intervention.

To administer transactions called with transactional RPCs effectively, you need to understand the states of transactions, how transactions are identified across servers, and how transactions use locking. For these transactions, you can display transaction information with Encina's **tkadmin** command, RRS ISPF panels, or subsystem commands specific to IMS or CICS. The following sections explain this.

Background on Encina Transaction Processing

When a transaction commits, all actions associated with that transaction are written to one or more logs by the coordinator, subordinate coordinator, and participating resource managers. For AS IMS or AS CICS transactions, the AS server entry in the RRS Unit of Recovery State log contains the participants in each AS transactional-RPC-initiated transaction. This includes the overall state of each transaction. For AS IMS, this also includes the associated IMS entry for each AS IMS transaction. The IMS or CICS log contains state information about each transaction involved in the global Encina transaction.

Notes:

1. The logs contain more than transactional information.
2. Transactions that are no longer active are not in the RRS Restart or Delayed Unit of Recovery State log or the IMS or CICS log.
3. Active transactions are in the active log and can be replayed.

The Two-Phase Commit Protocol

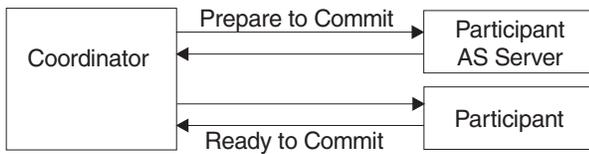
To synchronize related pieces of work taking place in different processes, distributed transactions use a two-phase commit protocol. The protocol guarantees that all the processes successfully complete the work or that it is not performed at all. For example, changes to data either all fail or all succeed. The goal is to ensure that each participant in a transaction, that is, each resource manager updating data in the transaction, takes the same action (commits or aborts).

Until a participant is prepared, it can unilaterally decide to abort the transaction. After all of the work of the transaction is complete, the application attempts to commit the work by invoking the two-phase commit protocol. In the first phase, the prepare phase, each participant is asked to prepare. When a participant is prepared, it agrees to accept whatever outcome the coordinator decides on (it can no longer unilaterally abort the transaction).

The final outcome of a transaction (how the transaction ends) depends upon the individual outcome of each participant in the transaction. In the resolution phase, a transaction commits if all participants can commit or aborts if any one participant aborts.

Figure 40 on page 140 shows an example in which AS is one of the participants in the Encina two-phase commit protocol.

Prepare Phase



Resolution Phase

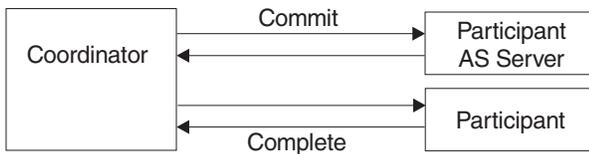


Figure 40. The Two-Phase Commit Protocol

- Encina selects one of the participating servers in the transaction to become the coordinator of the transaction. The coordinator tracks the status of the transaction and communicates with other participants in the transaction.
- Each participant is given an opportunity to prepare its work. Each participant notifies the coordinator that it is ready to commit or to abort. A participant can abort its work at any time before the prepare phase; however, once the participant “votes” to commit, it cannot abort.
- The coordinator tallies the results from all participants.
- The coordinator informs each participant of the result of the vote. The coordinator instructs each participant to commit if all participants have voted to commit or to abort if any one participant has aborted.
- Each participant takes the appropriate action.

The prepare phase consists of all actions up until the time each participant notifies the coordinator that it is ready to commit (or to abort). The resolution phase begins when the coordinator instructs each participant to either commit or abort, and ends when all participants have completed their work.

The States of an AS IMS or AS CICS Encina Transaction

When a client issues a transactional RPC to an AS server, several subsystems maintain transaction states and phases, including:

- Encina
- RRS
- IMS or CICS.

Each subsystem may have different names for each state of the transaction. See Table 20 on page 141 for an example of the differing names.

As shown in Table 20 on page 141, the states associated with the prepare phase are as follows:

- In the active state, a transaction may be accessing or modifying data.
- In the inactive state, a transaction may be waiting for input from a user.
- In the preparing state, a transaction may be preparing to notify a coordinator of its readiness to commit.

<i>Table 20. Phases and States of a Transaction</i>		
Encina Transaction State	Transaction State Associated with Transactions on RRS Log	Transaction State on IMS or CICS Log
Prepare Phase		
active	in_flight, in_statecheck	in_flight
inactive	in_flight	in_flight
preparing	in_prepare, (also for coordinator only, in_doubt, in_commit, or in_abort)	in_flight, in_doubt, or shunted (also, for coordinator only, in_commit or backout)
prepared	in_doubt	in_doubt
Resolution Phase		
committing or aborting	in_commit or in_backout	in_commit or in_abort
finished	forgotten, in_completion, in_forget	complete

Note: There are additional transaction states not described here. For information about all transaction states, see the **tkadmin list transaction** command in the Administrator's Guide for the Encina Toolkit for your system. For more information on the RRS states, see *z/OS MVS Programming: Resource Recovery*. For more information about the IMS states, see *IMS/ESA Operator's Reference*. For more information about CICS states, see the following publications:

- *CICS Supplied Transactions*
- *CICS System Programming Reference*
- *CICS Recovery and Restart Guide*.

After a transaction notifies a coordinator of its readiness to commit, the transaction enters the prepared state. The next phase is the resolution phase. In the resolution phase, a transaction can enter the states in the “committing” or “aborting” track, depending on the instruction from the coordinator. The final transaction state is finished. Note that because a transaction can abort at any time during the prepare phase, the states associated with aborting can occur during that phase. A transaction can also abort while in the active state.

Transaction Identifiers

Encina uses several identifiers to track transactions:

- Local transaction identifier (TID)** An identifier for a transaction at a given server. It is known only to that server.
- Global TID (GTID)** An identifier that ties a transaction to all of its participating applications across different servers. The GTID is unique across all servers.
- Unit of Recovery ID (URID)** A z/OS RRS identifier corresponding to the Encina GTID and TID for a transaction initiated through a transactional RPC. See *z/OS MVS Programming: Resource Recovery* for more information about RRS.

For an AS CICS server, RRS uses the following in addition to the URID.

- Unit of Work (UOW)** The UOW is 16 hexadecimal bytes. The first 8 bytes are the Application ID (APPLID). The second 8 bytes are a local Unit of Work ID.

When AS IMS is a part of the transaction, additional identifiers track transactions:

- Enterprise identifier (EID)** A z/OS Encina identifier uniquely identifying Encina's portion of the RRS URID. It consists of the Encina TID and GTID concatenated together. The first 4 bytes of the EID is the TID and the rest of the

EID is the GTID. IMS uses it to provide Encina information for an IMS transaction called with transactional RPC through AS IMS.

IMS-TOKEN

An IMS identifier uniquely identifying IMS's portion of the RRS URID. It is IMS's method in identifying a commit scope from one commit point to the next. It is 16 bytes long. The first 8 are the IMS subsystem name, and the last 8 consist of 2 hexadecimal counters, one a schedule count and the other a commit count within that schedule. The tokens are unique from one IMS cold start to the next.

IMS PSEUDO-TOKEN (P-TOKEN)

An IMS identifier available when in_doubt Units of Recovery (UORs) are identified in IMS. It is used in the IMS **DISPLAY UOR** and **CHANGE UOR** commands instead of the IMS-TOKEN. It also provides the URID and EID.

Because an application initiated with a transactional RPC to AS IMS or AS CICS is an Encina transaction, the Encina global identifier (GTID) is passed to the AS IMS server when the transaction is first initiated and a local TID is generated for each transaction. The Encina GTID is associated with the local TID. For active applications, **tkadmin list transactions** and **tkadmin query transaction** can provide information on the GTID and TID involved with transactions at a given server. For more information on these commands, see the Administrator's Guide for the Encina Toolkit for your system.

Additionally, because AS CICS or AS IMS and IMS themselves use RRS, the Encina transaction is also assigned a Unit of Recovery Identifier (URID), which is displayed when you query the GTID or local TID from an RRS Unit of Recovery ISPF panel.

When transactions are active in IMS as the result of an AS IMS call started through a transactional RPC, the IMS **DISPLAY UOR** command provides the IMS-TOKEN, URID, and EID as part of its output. If the IMS UOR is in an in_doubt state, the command also provides the IMS-PTOKEN. See *z/OS MVS Programming: Resource Recovery* and *IMS/ESA Operator's Reference* for more information. All these identifiers—local TID, GTID, URID, and IMS-TOKEN or IMS-PTOKEN—are needed if an administrator manually intervenes on or queries transactions. See “Viewing AS Server Encina Transaction Information” on page 144 for a discussion on using these identifiers to query or manually intervene in transactions.

Figure 41 on page 143 is an IMS example showing use of these identifiers. For this example, some identifiers are shortened from what you see when entering actual commands or panel queries.

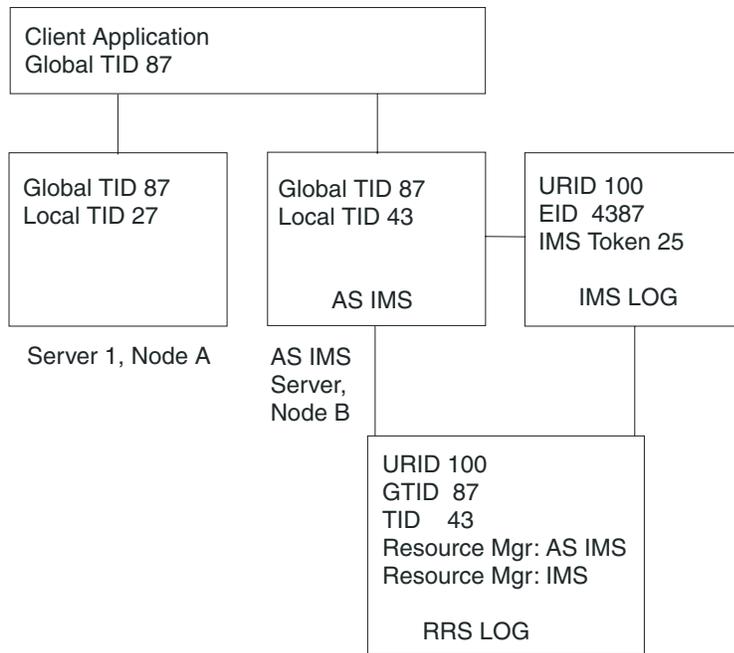


Figure 41. Identifiers for Encina Transactions Using AS IMS

The following items apply to the preceding figure:

- The client application with an Encina Application ID of 6 originates a transaction and makes a transactional RPC call to Server 1, Node A, and AS IMS server, Node B. The GTID for the transaction is 87. Server 1 and the AS IMS server are participants in the transaction.
- In server 1 on Node A, the transaction associated with GTID 87 is known as local TID 27.
- In AS IMS server on Node B, the transaction associated with GTID 87 is known as local TID 43.
- The AS IMS server calls an IMS transaction as a result of the client application's transactional RPC.
- AS IMS uses RRS for its logging and two-phase commit coordination with IMS. (AS IMS behaves as a coordinator to RRS.) AS IMS gives RRS the transaction's Encina GTID and TID, which RRS then associates with its RRS identity, the URID. IMS refers to this TID/GTID combination as the Enterprise Identifier or EID. (See the *z/OS MVS Programming: Resource Recovery* for more information about RRS.) In Figure 41, the transaction with Global TID 87 and Local TID 43 is known to RRS as URID 100.
- When AS IMS calls the transaction it must tell IMS that this is transactional work. When IMS processes the transaction, its work becomes part of the URID and IMS creates an IMS-TOKEN. Because RRS also knows about the Encina GTID and TID associated with an RRS URID, IMS obtains this information from RRS and saves it as an EID with its IMS-TOKEN information. In Figure 41, the transaction with Global TID 87 and Local TID 43 is known to IMS as IMS-TOKEN 25, which also is part of URID 100 and has an EID of 4387 (that is, a TID of 43 and GTID of 87). In IMS transaction recovery, all recoverable work is identified with its own recovery token. (If the IMS UOR was in an `in_doubt` state, there would also be an IMS P-TOKEN. The IMS P-TOKEN is needed when resolving `in_doubts` through the IMS **CHANGE UOR** command.)
- As shown in Figure 41, after IMS expresses interest in the transaction with RRS, RRS resource managers associated with URID 100 are AS IMS and IMS.

When you are using transactional RPCs from an AS CICS server, **CEMT CICS INQUIRE UOW** shows the state of a UOW (for example, `in_flight`) and whether it is active, waiting, or shunted. In a problematic transaction, you can use **CEMT SET UOW** to determine the outcome (backout, commit, or force) if the

transaction is shunted. You can force an `in_flight` transaction to abort by using **CEMT EXEC CICS SET TASK**. You can alter an `in_flight` transaction by using **tkadmin** commands.

Transactions and Locking in IMS

Encina transactions use locking and IMS uses locking on behalf of the calling application. Locking enables a transaction to restrict access to resources on which it depends. This ensures that two or more transactions cannot change the same data at the same time. Locking is important to guarantee the isolation property of transactions.

If two IMS transactions on the same IMS cause resource contention, IMS deadlock detection occurs. However, Encina applications running as a part of the global transaction but running on non-z/OS platforms also obtain locks, and conflicts in these applications can postpone the resolution of the IMS transaction. For more information on Encina locking and administering Encina transactions, see the Administrator's Guide for the Encina Toolkit for your system. When you use transactional RPC to update IMS resources, IMS holds all associated locks and message regions until the distributed transaction commits or aborts.

Viewing AS Server Encina Transaction Information

The administrator must know the names of the AS servers on the z/OS system or systems being administered and the DCE cell in which they are enrolled. If a CDS entry is set up for the AS servers (see Chapter 11, “Configuring the CDS Namespace Entries” on page 63), then after logging into the DCE cell where the AS servers have accounts, you enter:

```
cdscp list object /./AS/*
```

This command returns all the names of any AS servers defined and operational in this cell. The following is an example of output you might receive.

```
LIST
OBJECT  /.../dcecell.anyco.com/AS
AT      1997-02-24-15:48:35

merch1
merch2
merch3
```

The preceding shows that there are three AS servers, **merch1**, **merch2**, and **merch3**, that are defined and operational in the cell.

If you are using a different CDS configuration call **cdscp** specifying the alternate CDS entry. If you know the name or names of the AS servers, you can skip this step. For more information about the **cdscp** command, see the *z/OS DCE Command Reference*.

After you know the name or names of the AS servers, from a non-z/OS system where Encina clients are accessing AS servers, you can enter Encina administrator commands against the AS server. (Note that you need appropriate permissions to issue these commands against an AS server. See “Access to the Application Support Server Namespace Entries” on page 64 for more information.)

AS IMS Examples: For example, suppose `./AS/merch1` is an AS IMS server defined and operational in the cell. On the non-z/OS system, the administrator can enter:

```
tkadmin list transactions -server /./AS/merch1
```

This command returns a list of transactions (the local TIDs) and the current states of each transaction. For example:

65536 preparing

The output shows that there is one transaction at **./:AS/merch1** with a TID of 65536 and that its state is preparing.

After getting a list of transactions, you may want more information on one or more of them. In the preceding example, because transaction 65536 has a state of preparing, you might want more information about this transaction. You can enter:

```
tkadmin query transaction -server ./:AS/merch1 65536
```

The output from this command provides the current state of transaction 65536 (this number is the TID of the participating application in the transaction), the Application ID of the originating transaction, and the GTID. An example of output from this command is:

```
Global identifier:    00010000011201019868130a728ed011bddb02608c2f4727
State: preparing
Originator:         01019868130a728ed011bddb02608c2f4727
Participants:      01019868130a728ed011bddb02608c2f4728 01019868130a728ed011bddb02608c2f4729
```

For more information on the **tkadmin** command formats, see the Administrator's Guide for the Encina Toolkit for your system.

Knowing the Encina state, GTID, and TID of the active transaction lets you find out more information about the transaction in RRS or in IMS.

In RRS, using the RRS Unit of Recovery Selection ISPF panel, specify the GTID of the transaction to narrow the selection. Then specify the GTID of **00010000011201019868130a728ed011bddb02608c2f4727**. This returns the URID and the RRS state of all participants.

From the query panel, you can get more detailed information, including the name of the IMS involved in the transaction.

You can then access the IMS Resource Manager master console for the IMS involved and enter **DISPLAY UOR ALL**. This returns information about IMS units of recovery (UORs) for protected resources on the RRS/MVS recovery platform. Using the URID (from the RRS Unit of Recovery Selection panel obtained in the preceding) and the Encina TID and GTID (obtained previously by entering **tkadmin** commands), look at the output from the IMS **DISPLAY UOR ALL** command. You should find an entry that contains the specific URID, an EID (with the specific concatenated Encina TID and GTID), and the IMS-TOKEN for this IMS entry. You then have the RRS and IMS entries associated with a given AS IMS transaction. See *z/OS MVS Programming: Resource Recovery and IMS/ESA Operator's Reference* for more information on how to get status on a transaction when you know the Encina GTID and local TID.

If there are other AS IMS servers involved in the transaction and you need to find out the state of the transaction at another server, repeat the steps discussed previously, making sure you specify the GTID on the **tkadmin list transactions** command. For example, if the AS IMS server **./:AS/merch2** was involved in the transaction and you already know the GTID, you can enter:

```
tkadmin list transactions -server ./:AS/merch2 -global 00010000011201019868130a728ed011bddb02608c2f4727
```

The output would return the TID of the transaction at that server. You can then continue, using the commands and panels described previously.

If there are other non-AS IMS Encina servers involved in the transaction and you need to find out the state of the transaction at that server or servers, see the Administrator's Guide for the Encina Toolkit for your system.

AS CICS Transaction Examples: For example, suppose `./AS/mmb/DCEFVT5/cics2` is an AS CICS server defined and operational in the cell. On the non-z/OS system, the cell administrator can enter:

```
tkadmin list transactions -server ./AS/mmb/DCEFVT5/cics2
```

This command returns a list of transactions and local TIDs and the current state of each transaction. For example:

```
3 preparing
```

This output shows that the AS CICS server has one transaction with a TID of 3 and that its state is preparing. After getting the list of transactions, you may want more information on one or more of them. To get more information on TID 3, you can enter:

```
tkadmin query transaction -server ./AS/mmb/DCEFVT5/cics2 3
```

The output from this command provides the current state of transaction 3 (this number is the TID of the participating application in the transaction), the Application ID of the originating transaction, and the GTID. An example of the output from the command is:

```
Global identifier:    000100000112010108fefbc4fb90d211a43808005acd34e8
State: preparing
Originator:         010108fefbc4fb90d211a43808005acd34e8
Participants:      01010852bd1265467a1194af001234567890
                   010108fefbc4fb90d211a43808005acd34e8
```

For more information on the **tkadmin** command, see the Administrator's Guide for the Encina Toolkit for your system. Knowing the Encina state, GTID, and TID of the active transaction lets you find out more information about the transaction in RRS or CICS.

Using the RRS Unit of Recovery panel, specify the GTID or the TID to narrow the selection. To continue the preceding example, if you specify the TID of 00000003, this returns the URID and the RRS state of all the participants.

From the query panel, you can get more detailed information, including the resource manager name of the CICS involved in the transaction. From the CICS master console you can issue **CEMT INQ UOW**. This returns information about all the units of work on the system. Using the URID found in the RRS Query, find the UOW for the transaction. You should find an entry that has the first 8 bytes of the URID to display the CICS information on the transaction. You have the RRS and CICS entries associated with a given CICS transaction. See the *z/OS MVS Programming: Resource Recovery and CICS Supplied Transactions* for more information on how to get the status of a transaction when you know the TID and GTID.

Administering Transactions

When an application starts a transaction, the transaction should complete automatically when the application commits or aborts it. Administrators do not usually need to intervene in transactions. However, before or after the application commits or aborts the transaction, problems can occur outside of the application itself, denying access to the data involved in the transaction to other applications for long periods of time. Network or server outages can cause these problems. If any AS servers, RRS, IMS, CICS, or other Encina servers fail, they should be restarted. (If RRS fails, AS IMS or AS CICS also eventually fails as a result; in this case, restart RRS first.) If there are network outages, they should be corrected. Restarting the server allows Encina to automatically complete the two-phase commit processing associated with transactions being processed at the time of the server failure.

However, there could be a time when networks cannot be connected in a timely manner and you need to free IMS or CICS locks held on data. It is also possible that RRS, IMS, or CICS was cold-started, causing orphaned entries in RRS, IMS, or CICS that must be resolved. For these cases or any others where the transaction is problematic (that is, the transaction is not completing and appears to hang), you may need to intervene. “Intervening in Problematic Transactions” discusses actions you can take if you need to intervene in these situations.

Intervening in Problematic Transactions

Remember: Use manual intervention only rarely and with care. When at all possible, fix the problem or the failure and let Encina complete its two-phase commit processing. You can introduce inconsistencies into server data by forcing a transaction to the wrong outcome.

However, if you feel that manual intervention is unavoidable, perform the following steps to intervene in a problematic transaction.

1. Determine where the problem is:

- Network problems
- Server outages
- RRS restarted with cold log

To determine whether there is a network problem or server outage, enter **tkadmin** commands against all servers involved in the problematic transaction, being careful not to enter the **tkadmin** command from the same system where the transaction is running if you suspect network problems. See “Viewing AS Server Encina Transaction Information” on page 144 for a discussion of the **tkadmin** commands to enter and information returned. The **tkadmin** commands fail when there is a network or server outage. Use this failure to diagnose what and where the problems are.

If you restarted RRS with a cold log, there may be IMS log entries or CICS UOWs that are in_doubt, waiting for RRS to tell IMS or CICS to commit or abort. For cold log start situations, see “Cold Log Starts” on page 152.

2. Determine the best way to resolve the problem:

- If there are network problems, try to get the network operational so that Encina can complete the two-phase commit flows.
- If servers are down, restart them if possible, and let Encina recovery automatically complete the two-phase commit, thus freeing resource manager locks.
- If RRS was restarted with a cold log, manual intervention may be necessary in IMS or CICS. See “Cold Log Starts” on page 152 for more information.
- If you cannot restart the network or failed server, you need to determine the outcome of the transaction. See “Evaluating Transactions” on page 149 for a discussion on how to do this.

3. After you determine manual intervention is necessary and have determined the manual outcome, enter the necessary commands at the server or servers to resolve the problem. **Remember, however, you can introduce inconsistencies into server data by forcing a transaction to the wrong outcome.**

For IMS, before you can commit or abort the transaction, you need to know the server name(s), the GTID, TID, and, for an AS IMS server, the RRS URID and IMS P-TOKEN. For CICS, you need to know the UOW. (See “Transaction Identifiers” on page 141 for descriptions of these identifiers.) This information is already available to you because it was needed or discovered during your evaluation of the transaction.

- If you decide to abort the transaction at the AS server and the transaction state is active or in_flight, enter:

```
tkadmin abort transaction -server servername local_TID
```

For example, if an AS server, **./AS/merch1**, has a TID of 65536 and is active, you can enter:

```
tkadmin abort transaction -server ./AS/merch1 65536
```

If the command is successful, there is no output from the command.

If you decide to abort the transaction at the AS server and the transaction state is prepared, preparing, or in_doubt, enter:

```
tkadmin force transaction -server servername local_TID
```

For example, if an AS server, **./AS/merch1**, has a TID of 65536 and is in_doubt, you can enter:

```
tkadmin force transaction -server ./AS/merch1 65536
```

If the command is successful, there is no output from the command.

Either command aborts work done through the AS server in IMS or CICS. If there were other participants in the transaction, you may need to use one of these commands against each server involved in the GTID to abort all participants. For more information about the Encina **tkadmin** command, see the Administrator's Guide for the Encina Toolkit for your system.

If the network between the client and the AS server is not operational or if the AS server is not operational, you can abort the in_doubt transaction by using the RRS Unit of Recovery List ISPF panel. (See the *z/OS MVS Programming: Resource Recovery* for more information.) If RRS is not operational:

- For IMS use the IMS **CHANGE UOR** command. (See *IMS/ESA Operator's Reference* for more information.)
 - For CICS use the **CEMT SET TASK** command or **EXEC CICS SET TASK** to force an in_flight transaction to abort. Use the **CEMT SET UOW** command or **EXEC CICS SET UOW** to force a shunted transaction. See *CICS Supplied Transactions* for details.
- If you decide to commit the transaction, for the AS server enter:

```
tkadmin force transaction -server servername local_TID -commitdesired
```

For example, if an AS IMS server, **./AS/merch1**, has a TID of 65536 and is in_doubt, you can enter:

```
tkadmin force transaction -server ./AS/merch1 65536
```

If the command is successful, there is no output from the command.

This command commits work done through the AS IMS server in IMS. If there were other participants in the transaction, you may need to use this command against each server involved in the GTID to commit all participants. For more information about the Encina **tkadmin** command, see the Administrator's Guide for the Encina Toolkit for your system.

If the network between the client and the AS server is not operational or the AS server is not operational, you can commit the transaction by using the RRS Unit of Recovery List ISPF panel. (See *z/OS MVS Programming: Resource Recovery* for more information.) If RRS is not operational:

- For IMS use the IMS **CHANGE UOR** command. (See *IMS/ESA Operator's Reference* for more information.)
- For CICS use the **CEMT SET UOW** command if the transaction's state is shunted.

Evaluating Transactions: You should force an outcome only in rare cases and as a last resort because you can introduce inconsistencies into the data of the server if you force the wrong outcome. However, if you need to manually intervene in a problematic transaction, you need to determine the state of the transaction before you can determine whether to force an AS IMS or CICS transaction to commit or to abort. (There may be more than one transaction that is having problems at an AS server. Perform the following steps for each.)

- Review the transaction state in each of the transaction's participating applications. In the AS server, you should determine the state of the transaction in Encina, RRS, and IMS or CICS, if possible. Start with the AS servers involved in the transaction or transactions. See “Viewing AS Server Encina Transaction Information” on page 144 for a discussion of locating AS servers and the Encina **tkadmin** commands, RRS ISPF panels, and IMS or CICS commands to use to get the current state of the transaction in each.

You may not be able to enter all the commands or may need to use additional RRS ISPF panels because of network outages or failed servers. For example, suppose you had the situation shown in Figure 42.

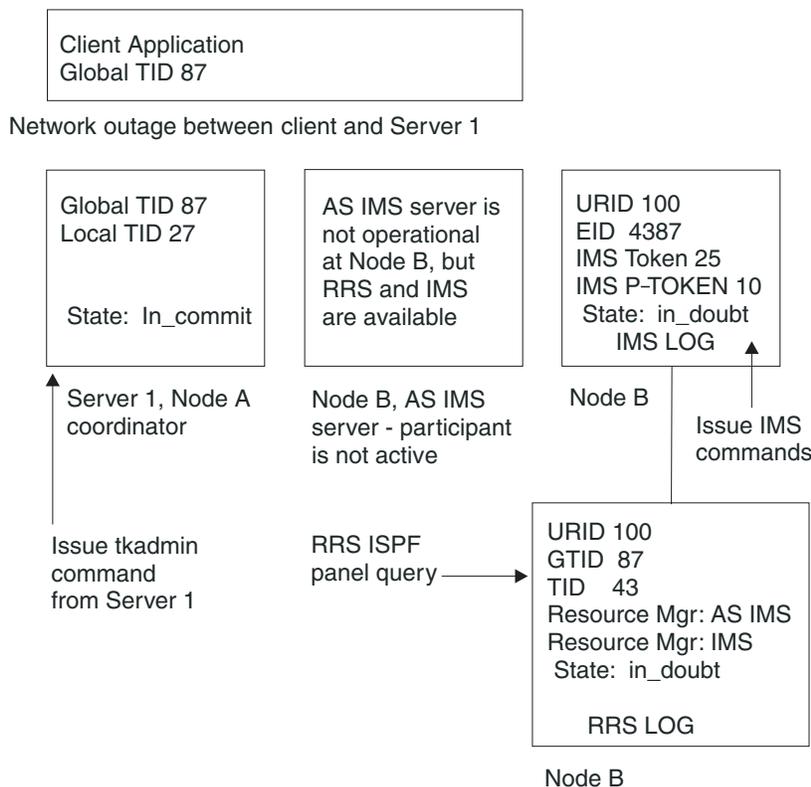


Figure 42. AS IMS Failure Scenario Example for Encina Transaction

You enter **tkadmin** commands from the client application machine to the AS server and discover that AS IMS is not operational. In this example, you need to get a list of all URIDs that AS IMS was processing at the time it failed. To do this you need to know the name by which RRS knows the AS server. This is specified in the `_ASU_RESOURCE_MGR_NAME` environment variable for that AS server. Access the RRS Resource Manager Selection ISPF panel and provide that Resource Manager name. From the output returned, you can get to the RRS Unit of Recovery Selection ISPF panel. You are returned a list of URIDs and their RRS states that AS was processing when it failed. For every entry on the list, you can ask for more detail. In this example, assume there was one entry. The RRS Unit of Recovery Details ISPF panel provides, among other information, the GTID, TID, and IMS Resource Manager involved in the transaction and the RRS state of the transaction. In this example, the state would be `in_doubt`, the GTID is 87, the TID is 43, and the URID is 100. A state of

in_doubt says that the IMS transaction can be told to Commit or Abort when AS IMS eventually comes up, but you do not know which it is. You need to see if you can find out the state of the transaction at the other server involved in the transaction.

You enter the **tkadmin** commands from the client application machine to Server 1. However, the network connection between the client application machine and Server 1 is not operational. You then call the **tkadmin** commands from the machine where Server 1 is operational. Server 1 is up and running. Because it is the coordinator of the two-phase commit processing, it knows the outcome. It returns information to you for the transaction with a GTID of 87; the State is In_Commit. After you know that the state of the transaction is Commit, you can safely use the RRS Unit of Recovery List ISPF panel for the URIDs that the AS server was processing. For URID 100, you previously determined that it had a GTID of 87. You can commit URID 100. This causes IMS to commit its work for URID 100. You have freed the IMS lock on the data involved with RRS URID 100. The RRS entry for URID 100 remains until the AS IMS server restarts and does recovery processing with RRS.

As you can see from the preceding AS IMS example, determining the state of a problematic transaction involves knowing the servers involved in a transaction and sometimes the system on which they are running. There may be many ways of determining the state of a failed transaction, although the preceding discussed only one. The important information that you ultimately need to obtain is a transaction state that does not cause data inconsistencies if you manually commit or abort the transaction.

- Determine the action to be performed from the transaction state.

<i>Table 21. Action to Perform Based on Transaction State</i>			
Encina Transaction Coordinator State	RRS State	IMS/CICS State	Action to Perform
active	in_flight	in_flight	See “Intervening in Unprepared Transactions” on page 152 and then manually abort the transaction.
inactive	in_flight	in_flight	See “Intervening in Unprepared Transactions” on page 152 and then manually abort the transaction.
preparing	in_prepare	in_flight	See “Intervening in Unprepared Transactions” on page 152 and then manually abort the transaction.
		in_doubt (IMS) or shunted (CICS)	See “Intervening in Unprepared Transactions” on page 152 and then manually abort the transaction.
	backout	in_doubt (IMS) or shunted (CICS)	See “Intervening in Unprepared Transactions” on page 152 and then manually abort the transaction.
	in_doubt	in_doubt (IMS) or shunted (CICS)	NO manual action is safe to perform. Before performing any manual action, see “Intervening in Unprepared Transactions” on page 152.
prepared	in_commit	in_doubt	See “Intervening in Unprepared Transactions” on page 152 and then commit the transaction.
	in_doubt	in_doubt	NO manual action is safe to perform. Before performing any manual action, see “Intervening in Unprepared Transactions” on page 152.
committing	in_doubt	in_doubt	First, see “Intervening in Unprepared Transactions” on page 152. Then commit the transaction.
	in_commit	in_doubt	First, see “Intervening in Unprepared Transactions” on page 152. Then commit the transaction.
aborting	in_doubt	in_doubt	Abort the transaction.
	backout	in_doubt	Abort the transaction.
finished	forgotten, in_completion, or in_forget	committed	There is NO manual action to perform.

- If you cannot determine a safe outcome for a transaction, examine locks. Conflicting locks can cause a hung transaction, which may cause the need for manual intervention. However, if a safe outcome cannot be determined for a transaction and there are no lock conflicts, avoid manual intervention. For more information about investigating locks on these systems, see the *IMS/ESA Operator's Reference*. For more information about the transactional state for CICS, see *CICS Supplied Transactions* and *CICS System Programming Reference*. You may also want to see the Administrator's Guide for the Encina Toolkit for your system.

Intervening in Unprepared Transactions: An unprepared transaction can be aborted at any time. An unprepared transaction can be:

- Unprepared Encina transactions
- Unprepared RRS transactions
- Unprepared IMS or CICS transactions

Unprepared Encina transaction states include: active, inactive, and preparing. Unprepared RRS transaction states include in_flight and in_doubt. Unprepared IMS transaction states include in_flight and in_doubt. An unprepared CICS transaction state is in_flight.

Encina uses a presume-abort transaction protocol. Therefore, aborting an unprepared transaction does not affect the consistency of data because all the transaction's participants have not yet notified the coordinator of their readiness to commit. If you abort an unprepared transaction, any work that was completed on behalf of the transaction is rolled back.

When you abort a transaction, the aborted transaction releases locks that other transactions may be waiting for, possibly freeing a server to perform other operations. However, any work the transaction did must be redone. You must weigh the impact of aborting the transaction. Aborting the transactions can result in better performance by releasing locks and allowing other transactions to run but with a penalty of later rework.

Intervening in Prepared Transactions: With transactions that have already prepared, any action that you take is more significant. To release the locks held by a prepared transaction, you must force it to commit or to abort. Forcing might be required in the following situations:

- When a transaction can never be completed, for example, if a server crashes and has to be cold started, thus losing the transaction log. For an AS server, this could be the cold start of RRS.
- When critical resources are held, for example, if RRS is down but IMS or CICS is still operational.
- When system performance suffers, for example, all servers involved in a transaction are operational, but there are network problems.

If you do not know the coordinator decision, you can introduce data inconsistencies whether you force a transaction to commit or to abort. Forcing a transaction to abort may result in better damage reporting by Encina to other applications.

If the final transaction state is known on other servers, the transaction should be resolved the same way on the problem system.

Cold Log Starts

Remember that the coordinator and participant log information. These logs may reside in RRS, IMS or CICS, or any other server involved in the transaction. If any servers fail during the two-phase commit processing and are restarted with a cold log, orphaned entries can remain on the logs of the other servers. The following discusses implications of restarting servers involved in Encina transactions with cold logs.

- AS Server

Application Support logs its information in RRS. RRS associates this log information to a specific server through the value in ASU_RESOURCE_MGR_NAME. If an AS server changes its name or if the RRS logs associated with the names are lost, manual intervention is needed to resolve any incomplete transaction. To resolve such transactions, you need to know the name by which RRS previously knew the AS server. Then access the RRS Resource Manger Selection ISPF panel and provide that Resource Manager name. From the output returned you can get to the RRS Unit of Recovery Selection ISPF panel. You are returned a list of URIDs the AS server was processing

before its failure and their RRS states. After you get this information, you can determine the states of other resource managers (such as CICS or IMS) that were involved in a URID. If the state is in_commit or in_backout, AS could have been acting as the Encina coordinator. If it is in_doubt, you need to determine the outcome of the transaction. Before you can manually remove these entries, you must query any other servers that were participants in the transaction and determine the transaction state at that server.

See “Viewing AS Server Encina Transaction Information” on page 144 and “Evaluating Transactions” on page 149 for more information on determining the state of other participants. Once the state is determined, manually commit, abort or remove the entries at each server involved in the transaction. See “Intervening in Problematic Transactions” on page 147 for more information.

- RRS

If RRS is cold started—that is, the RRS Resource Manager (RM) log—any IMS or CICS UOWs are orphaned that are in the in_doubt state and that had expressed interest to RRS for transactions started with transactional RPCs and an AS server. Manual intervention is needed to remove these entries from the list. To handle orphaned UOWs in IMS, find the EID for each entry by using the IMS command **DISPLAY UOR ALL**. Obtain the GTID from the EID; (remember the first 4 bytes of the ETID is the TID and the rest is the GTID). Specify the GTID in the appropriate Encina **tkadmin** command against the other server involved in the transaction to determine the correct manual action to take. See “Viewing AS Server Encina Transaction Information” on page 144, “Intervening in Problematic Transactions” on page 147, and “Evaluating Transactions” on page 149 for more information.

To handle orphaned UOWs in CICS, enter the CICS transaction **CEMT INQ UOW**. Set the UOWSTATE to COMMIT or BACKOUT for all UOWs that include all of the following:

- UOWSTATE(INDOUBT)
- WAITSTATE(SHUNTED)
- WAITCAUSE(RRMS)

Note: Until you know RRS has been cold-started, do not force the UOW to a commit or backout state. (A warm start of RRS automatically resolves the UOWs.)

- IMS or CICS

If IMS or CICS was cold started, RRS may still have entries for transactions started through an AS server for which IMS or CICS had expressed an interest. When IMS or CICS restarts, RRS tries to complete the two-phase commit processing for those entries. Depending on the state, the entries may remain after this attempt. You can manually remove these IMS or CICS entries from RRS by using the RRS Unit of Recovery List ISPF panel. See *z/OS MVS Programming: Resource Recovery* for more information. For more information on IMS cold starts, see the *IMS/ESA Operations Guide*. For more information on CICS cold starts, see the *CICS Recovery and Restart Guide*.

- Other Encina Servers

If another Encina server involved in a transaction with an AS server fails and is restarted with a cold log, it can cause orphaned entries for the AS server in RRS. You need to intervene manually to determine the outcome and remove these entries. See “Viewing AS Server Encina Transaction Information” on page 144, “Intervening in Problematic Transactions” on page 147, and “Evaluating Transactions” on page 149 for a discussion on how to do this.

Chapter 22. Problem Determination

z/OS DCE provides a messaging facility that displays messages from DCE services or DCE applications. You can control the display of these messages based on the severity level using the `_EUV_SVC_MSG_LEVEL` environment variable. You can also control where the messages are displayed using the `_EUV_SVC_MSG_LOGGING` environment variable.

You can set these environment variables in the Application Support server's `envar` file. Chapter 4, "Creating the Environment Variable File" on page 21 discusses the `envar` file.

This chapter describes the messaging facility in z/OS DCE and the Application Support server-specific aspects of the generated message log.

Setting `_EUV_SVC_MSG_LEVEL`

The `_EUV_SVC_MSG_LEVEL` environment variable sets a minimum level of error severity at which messages are displayed.

Following are the possible values of the `_EUV_SVC_MSG_LEVEL` environment variable:

NONE	No messages are logged.
FATAL	Only fatal messages are logged. Usually, these messages are issued when a certain degree of permanent loss or damage occurs, such as the corruption of the database.
ERROR	Only error and fatal messages are logged. Error messages are issued when an unexpected event occurs that is recoverable or can be corrected by manual intervention.
USER	Only user, error, and fatal messages are logged. User messages are issued when an error occurs on the use of the DCE Application Programming Interfaces (APIs).
WARNING	Only warning, user, error, and fatal messages are logged. Warning messages are issued when any of the following occurs: <ul style="list-style-type: none">• An error occurred but was automatically corrected by the program or system.• A condition was detected that may be an error, depending on whether the effects of the condition are acceptable.• A condition exists that if left uncorrected will eventually result in an error.
NOTICE	Only key informational messages are logged, along with warning, error, user, and fatal messages. Notice messages are generally for major events such as the startup of a server.
VERBOSE	All messages are logged. This is the default action. VERBOSE messages include informational messages on events which are important in monitoring DCE, such as the creation and deletion of RPC endpoints.

Setting `_EUV_SVC_MSG_LOGGING`

You can also control where the messages are displayed, using the `_EUV_SVC_MSG_LOGGING` environment variable. You can set this variable to any of the following values:

NO_LOGGING	All messages are suppressed.
STDIO_LOGGING	Notice, verbose, and warning messages are routed to the standard output file (the screen or the standard output DD statement). All other messages are routed to the standard error file (the screen of the standard error DD statement). This is the default value.
CONSOLE_LOGGING	Key informational messages and all error, notice, and fatal messages are routed to the operator console. All other messages are routed to the standard output file (the screen or the standard output DD statement). This is the recommended value for this variable.

Error Correlator ID

For AS IMS with ISC only, some messages logged for the Application Support server contain a **correlator ID**. The correlator ID is returned to the client so that the ASUADMIN message related to the client's error condition can be located. Based on the correlator ID, the administrator can determine the proper messages in the message log.

For example, any client who makes a request to the Application Support server receives a specific correlator ID. If an abnormal event occurs, the client gives the correlator ID to the administrator. The administrator then views the message log and searches for all messages corresponding to the specific correlator ID. The administrator is able to determine the cause of the abnormal event.

Using Message ASUV430I for Problem Solving in AS IMS

The message ASUV430I allows the DCE AS IMS administrator to relate a DCE client with a specific IMS ISC session. This message gives the node and the subpool that is opened for the client. Based on this information, the administrator can perform problem determination using IMS facilities.

In order to generate this information, the `_EUV_SVC_MSG_LEVEL` environment variable must be set to at least NOTICE.

Using the ASUVPLOG Data Set to Capture IMS Messages

For AS IMS with ISC only, the ASUVPLOG data set is used to capture unsolicited IMS messages sent to one of the Application Support server ISC sessions. This may have come as a result of IMS messages left in the output queue from a previous session failure. These unsolicited messages may also have been due to asynchronous messages sent to the ISC sessions during normal transaction execution.

The log is identified in the startup PROC by an ASUVPLOG DD statement.

Allocating the ASUVPLOG Data Set

The ASUVPLOG data set must be allocated as a sequential disk data set, with enough space to hold the data until it can be reviewed. In some installations, it may need as little as one cylinder of DASD space. Select a size that is reasonable for your environment, monitor its use, and re-allocate with a different size if necessary at a later time.

Every time the attachment is started, a header record of dashes with an embedded time stamp is written, and the data set wraps when it is full. No DCB attributes need to be provided because the Application Support server uses it own.

The administrator can also send the log to SYSOUT. In this case, the text does not wrap.

The format of a line in the log is as follows:

timestamp:	uuid:	hexadecimal data:	text:
a688deaa274d3000	2CD6F160-0903-1AF5-AE2F-C9C2D4FF00A2	0805020201020001 0066f1f1f1f1f1f1	'.....111111'
a688deaa27997a00	2CD6F160-0903-1AF5-AE2F-C9C2D4FF00A2	f1f1f1f1f1f1f1f1 f1f1f1f1f1f1f1f1	'1111111111111111'
a688deaa27c33f00	2CD6F160-0903-1AF5-AE2F-C9C2D4FF00A2	f1f1f1f1f1f1f1f1 f1f1f1f1f1f1f1f1	'1111111111111111'
a688deaa27ef6900	2CD6F160-0903-1AF5-AE2F-C9C2D4FF00A2	f1f1f1f1f1f1f1f1 f1f1f1f1f1f1f1f1	'1111111111111111'

The time stamp is helpful in determining the sequence in which the lines appeared in the file.

The UUID is the principal UUID of the client (which has the ISC session allocated) for which the message is intended. The hexadecimal data follows, 16 bytes per line. Any printable text is also printed. Periods replace nonprintable characters. A line of dashes shows the location of the current end of file.

Appendix A. ASULUEE0 Exit Routine

ASULUEE0 is the System/370 assembler source code for the DFSLUEE0 exit. ASULUEE0 must be assembled and link-edited as module DFSLUEE0. The DFSLUEE0 exit contains the logic that allows Application Support to be able accept and understand MOD Name and LTERM data. See "Step 5: Configure the IMS DFSLUEE0 Exit Routine" on page 46 for information on configuring DFSLUEE0. The assembler source for this exit is supplied as member ASULUEE0 in the *ASPREFX.SASUSAMP* data set.

```
LUEE      TITLE 'LU 6.2 Input Output Edit Exit'                00010000
DFSLUEE0 CSECT                                                00020000
*****                                                        00030000
*                                                        00040000
* Description: LUM Services                                    00050000
*                                                        00060000
*****@SCPYRT**                                              00070000
*                                                        00080000
*      "Restricted Materials of IBM"                          00090000
*      Licensed Materials - Property of IBM                   00100000
*                                                        00110000
*      5685-013 (C) Copyright IBM Corporation 1991            00120000
*      5647-A01 (C) Copyright IBM Corporation 1995, 1997     00130000
*                                                        00140000
*****@ECPYRT**                                              00150000
*                                                        00160000
*****                                                        00170000
*                                                        00180000
* NOTE: The OS/390 DCE Application Support executable         00190000
* source statements supplied in this exit are for use with   00200000
* Executable source statements for use with IMS/ESA 5.1 or   00210000
* supplied as comments. You must make any appropriate changes 00220000
* to your installed IMS version before installing this exit   00230000
* routine.                                                    00240000
* For IMS/ESA 4.1, ensure statements (3) marked "@IMS5.1+"   00250000
* and statement (1) marked "@IMS4.1" is not a comment.      00260000
* For IMS/ESA 5.1+, ensure statement (1) marked "@IMS4.1"   00270000
* and statements (3) marked "@IMS5.1+" are not comments.     00280000
*                                                        00290000
* For any IMS version, customization of the table at label  00300000
* ASUJLTAB is mandatory if this exit is installed.           00310000
*                                                        00320000
*****                                                        00330000
*                                                        00340000
* status:      release 4.1      change level - 0              00350000
*                                                        00360000
* function:    this module is called by IMS                    00370000
*              prior to inserting a message                    00380000
*              segment to the message queue and                 00390000
*              before sending a message                          00400000
* dependency:  IMS/ESA 4.1 or higher                            00410000
*                                                        00420000
```

Figure 43 (Part 1 of 7). ASULUEE0 Exit Routine

```

* module type: procedure                                00430000
*   attribute: re-entrant                              00440000
*   processor: assembler h                             00450000
*                                                       00460000
* registers:   standard entry and exit                 00470000
*                                                       00480000
* module size: see assembler listing                  00490000
*                                                       00500000
* linkage:                                           00510000
*                                                       00520000
* entry point: DFSLUEE0                               00530000
*                                                       00540000
* input:                                             00550000
*   register 1 is pointing to a parameter area which contains the 00560000
*   following fields:                                00570000
*     word 1 - address of input output flag           00580000
*               flag = 0 means input message segment 00590000
*               flag = 4 means output message segment 00600000
*               this is an entry parameter only.     00610000
*     word 2 - pointer to input or output message segment 00620000
*               it is two bytes of length + two bytes of 00630000
*               flag field + the actual message      00640000
*               the exit routine can change the message 00650000
*               and contents, provided that it resets the 00660000
*               message length field to correspond with the 00670000
*               new length.                          00680000
*               the exit routine can increase the message 00690000
*               length up to 256 bytes, but the total message 00700000
*               length can't exceed 32,767 bytes including 00710000
*               message length and flag fields. if the 00720000
*               message exceeds this limit, IMS truncates 00730000
*               the message and issues DFS1967 to the current 00740000
*               MTO to indicate a message buffer overlay. 00750000
*               supported by IMS queue manager        00760000
*     word 3 - pointer to the transaction code (fixed length 00770000
*               left justified, padded with blanks)  00780000
*               this is an entry parameter only     00790000
*     word 4 - pointer to LU name (fixed length, left 00800000
*               justified, padded with blanks)      00810000
*               this is an entry parameter only     00820000
*     word 5 - pointer to user id (fixed length, left 00830000
*               justified, padded with blanks)      00840000
*               this is an entry parameter only     00850000
*     word 6 - pointer to a word which contains the 00860000
*               return code                          00870000
*               this is an exit parameter only.    00880000
*     word 7 - pointer to lterm (fixed length, left 00890000
*               justified, padded with blanks)      00900000
*               this is an exit parameter only     00910000

```

Figure 43 (Part 2 of 7). ASULUEE0 Exit Routine

```

*          word 8 - pointer to mod name (fixed length, left      00920000
*                    justified, padded with blanks)             00930000
*                    this is an exit/entry parameter           00940000
*          word 9 - pointer to user table                        00950000
*                    this is an entry parameter only           00960000
*          word 10- pointer to msg flag                         00970000
*                    bit0 = 1 means first segment              00980000
*                    this is an entry parameter only           00990000
* register 13 address of save area. the exit routine must not  01000000
*                    change the first three words.             01010000
*                                                            01020000
*          normal linkage conventions for register 13, 14 and 15 01030000
*                                                            01040000
* exit-normal :                                               01050000
*                                                            01060000
* return_code contains one of the following value:            01070000
* 0 - LU 6.2 support will continue processing                  01080000
* 4 - discard this message segment                            01090000
* 8 - deallocate_abend the conversation                       01100000
* IMS treats any other return codes as 0                     01110000
* normal linkage convention for registers 13, 14 and 15        01120000
*                                                            01130000
***** 01140000
          SAVE (14,12),,DFSLUEE0&SYSDATE&SYSTIME @BN40234 01150000
          LR R12,R15 @BN40234 01160000
          USING DFSLUEE0,R12 @BN40234 01170000
          USING PARM_LIST,R1 01180000
* 01190000
* Beginning of OS/390 DCE Application Support changes to      01200000
* default DFSLUEE0 exit routine as supplied by IMS/ESA. The next five 01210000
* lines of code shown as comments is the default DFSLUEE0 routine as 01220000
* shipped by IMS/ESA. 01230000
* 01240000
* L R2,PARM_RETURN_CODE get rc field addr 01250000
* LA R15,0 set r15 to 0 01260000
* ST R15,0(,R2) set 0 return code 01270000
* DROP R1 01280000
* RETURN (14,12) return to the caller @BN40234 01290000
* 01300000
          LR R10,R1 re-base parameter list 01310000
          DROP R1 01320000
          USING PARM_LIST,R10 01330000
          B ASUVJEE0 go to Application Support exit rtn 01340000
DFSLURC0 DS 0H common exit point 01350000
          L R2,PARM_RETURN_CODE get rc field addr 01360000
          LA R15,0 set r15 to 0 01370000
DFSLURCX DS 0H common exit point 01380000
          ST R15,0(0,R2) set return code 01390000
          RETURN (14,12) return to the caller @BN40234 01400000
          EJECT 01410000

```

Figure 43 (Part 3 of 7). ASULUEE0 Exit Routine

ASUVJEE0	DS	0H		01420000
	B	ASUJSTRT	branch around eye catchers	01430000
	DC	C'5647-A01 '		01440000
	DC	C'(C) Copyright IBM Corporation 1995, 1997. '		01450000
	DC	C'All rights reserved. '		01460000
	DC	C'Licensed Materials - Property of IBM. '		01470000
	DC	C'US Government Users Restricted Rights - '		01480000
	DC	C'Use, duplication or disclosure restricted by '		01490000
	DC	C'GSA ADP Schedule Contract with IBM Corp.'		01500000
	SPACE			01510000
ASUJSTRT	DS	0H	(determine if AS IMS is involved)	01520000
	L	R5,PARM_LUNAME	get address of LU name	01530000
	LA	R3,ASUJLTAB	get address of LU name table	01540000
	LA	R4,ASUJLTEN	get number of LU table entries	01550000
*	TRT	0(9,R5),ASUJTRTT	look for period in name @IMS5.1+	01560000
*	BZ	DFSLURC0	no hit in first 9 chars !? @IMS5.1+	01570000
ASUJSLOP	EQU	*		01580000
	CLC	0(8,R5),0(R3)	incoming LU name in table? @IMS4.1	01590000
*	CLC	1(8,R1),0(R3)	does LU name part match? @IMS5.1+	01600000
	BE	ASUJCKIN	yes, its ours	01610000
	LA	R3,ASUJLTEN(0,R3)	no, bump to next table entry	01620000
	BCT	R4,ASUJSLOP	loop thru table	01630000
	B	DFSLURC0	not found, exit with rc=0	01640000
	SPACE			01650000
ASUJCKIN	EQU	*	(check for input vs. output msg)	01660000
	L	R2,PARM_IOFLAG	get address of I/O flag	01670000
	ICM	R0,B'1111',0(R2)	is flag 0?	01680000
	BZ	ASUJINPM	yes, input message	01690000
	LA	R15,4	set up reg for compare to 4	01700000
	CLR	R0,R15	is flag 4?	01710000
	BE	ASUJOUTM	yes, output message	01720000
	B	DFSLURC0	something else, ignore for now	01730000
	SPACE			01740000
ASUJINPM	EQU	*	(process input message segment)	01750000
	L	R2,PARM_MSG_FLAG	point to message flag	01760000
	TM	0(R2),X'80'	is bit 0 on? (first msg segment)	01770000
	BNO	DFSLURC0	no, don't touch message	01780000
	L	R4,PARM_MESSAGE_SEG	get address of message segment	01790000
	USING	ASUJIMSG,R4		01800000
	CLC	ASUJIMLL,=AL2(L'ASUJCMSG+4)	length of AS IMS test cmd?	01810000
	BNE	ASUJINP0	no, try for lterm/modname segment	01820000
	CLC	ASUJITXC,ASUJCMSG	yes, is this the AS IMS test cmd?	01830000
	BNE	DFSLURC0	no, don't touch message	01840000
	MVC	ASUJITXD,ASUJCMS2	substitute /DISPLAY APPC command	01850000
	MVC	ASUJIMLL,=AL2(L'ASUJCMS2+4)	fudge the ll for new cmd	01860000
	B	DFSLURC0	normal exit (and pass cmd to IMS)	01870000
	SPACE			01880000

Figure 43 (Part 4 of 7). ASULUEE0 Exit Routine

ASUJINP0	EQU	*		01890000
	LH	R3,ASUJIMLL	get length of input message segment	01900000
	CH	R3,=AL2(ASUJIMLE)	at least long enough for AS IMS	01910000
*			lterm/MOD name segment?	01920000
	BL	DFSLURC0	no, don't touch message	01930000
	AR	R4,R3	point to end of message segment	01940000
	S	R4,=A(ASUJIMLD)	now back to start of data	01950000
	USING	ASUJIPRF,R4	move addressability	01960000
	CLC	ASUJIPRF,ASUJPRFX	is this the AS IMS signature?	01970000
	BNE	DFSLURC0	no, don't touch message	01980000
	CLC	ASUJIMLT,=CL8' '	is input LTERM blank?	01990000
	BE	ASUJINP1	yes, ignore it	02000000
	L	R2,PARM_LTERM	point to LTERM field	02010000
	MVC	0(L'ASUJIMLT,R2),ASUJIMLT	give LTERM to IMS	02020000
ASUJINP1	EQU	*		02030000
	CLC	ASUJIMMO,=CL8' '	is input MOD name blank?	02040000
	BE	ASUJINP2	yes, ignore it	02050000
	L	R2,PARM_MODNAME	point to MOD name field	02060000
	MVC	0(L'ASUJIMMO,R2),ASUJIMMO	give MOD name to IMS	02070000
ASUJINP2	EQU	*		02080000
	B	ASUJRC4	set rc=4 to discard this segment	02090000
	SPACE			02100000
ASUJOUTM	EQU	*	(process output message segment)	02110000
	L	R4,PARM_MESSAGE_SEG	get address of message segment	02120000
	USING	ASUJOMSG,R4		02130000
	LH	R3,ASUJOMLL	get length of message	02140000
	LTR	R3,R3	test it	02150000
	BNP	ASUJRC8	huh? bail out and deallocate conv	02160000
	LA	R5,8(0,R3)	account for length of MOD name	02170000
	CH	R5,=H'32767'	check if under max length	02180000
	BH	ASUJRC8	bad news, bail out	02190000
	L	R2,PARM_MODNAME	point to MOD name field	02200000
	AR	R3,R4	point to end of message	02210000
	MVC	0(8,R3),0(R2)	append MOD name to end of message	02220000
	STH	R5,ASUJOMLL	update length field in message	02230000
	B	DFSLURC0	normal exit	02240000
	DROP	R4		02250000
	SPACE			02260000
ASUJRC4	EQU	*		02270000
	L	R2,PARM_RETURN_CODE	get rc field addr	02280000
	LA	R15,4	set r15 to 4	02290000
	B	DFSLURCX	go to common exit point	02300000
	SPACE			02310000
ASUJRC8	EQU	*		02320000
	L	R2,PARM_RETURN_CODE	get rc field addr	02330000
	LA	R15,8	set r15 to 8	02340000
	B	DFSLURCX	go to common exit point	02350000
	DROP	R10		02360000
	SPACE			02370000
*				02380000

Figure 43 (Part 5 of 7). ASULUEE0 Exit Routine

```

* The ASUJLTAB table contains one entry for each LU name assigned to an 02390000
* Application Support IMS APPC server. This table is used by the 02400000
* ASUVJEE0 routine to determine if a message segment was sent by or is 02410000
* to be received by an AS IMS server. This table must be updated as 02420000
* part of the customization process for this exit routine. There MUST 02430000
* be at least ONE entry in this table! 02440000
* 02450000
ASUJLTAB EQU * 02460000
      DC CL8'MVSIMS01' sample name (change or remove) 02470000
      DC CL8'MVSIMS02' sample name (change or remove) 02480000
ASUJLTLN EQU 8 length of each table entry 02490000
ASUJLTEN EQU (*-ASUJLTAB)/ASUJLTLN number of entries in table 02500000
      SPACE 02510000
ASUJCMSG DC C'/DISPLAY MODNAME 00746A00 LTERM LUEETEST' AS IMS cmd 02520000
ASUJCMS2 DC C'/DISPLAY APPC' legitimate IMS command for AS IMS 02530000
ASUJPRFX DC C'ASUVINFO00746A00' lterm/MOD name segment prefix 02540000
      SPACE 02550000
ASUJTRTT DC 256X'00' TRT table 02560000
      ORG ASUJTRTT+C'.' 02570000
      DC X'FF' stop on period 02580000
      ORG , 02590000
      LTORG 02600000
      EJECT 02610000
ASUJIMSG DSECT input message segment 02620000
ASUJIMLL DS CL2 message 'll' field 02630000
ASUJIMZZ DS CL2 message 'zz' field 02640000
ASUJITXC DS 0CL(L'ASUJCMSG) input message text (AS IMS test cmd) 02650000
ASUJITXD DS 0CL(L'ASUJCMS2) altered msg text for legit IMS cmd 02660000
      DS 0C transaction code w/blnk appears here 02670000
ASUJIPRF DS CL(L'ASUJPRFX) lterm/MOD name segment prefix 02680000
ASUJIMLT DS CL8 LTERM name from AS IMS 02690000
ASUJIMMO DS CL8 MOD name from AS IMS 02700000
ASUJIMLE EQU *-ASUJIMSG length of lterm/modname msg segment 02710000
ASUJIMLD EQU *-ASUJIPRF length of lterm/modname msg data 02720000
      SPACE 02730000
ASUJOMSG DSECT output message segment 02740000
ASUJOMLL DS CL2 message 'll' field 02750000
ASUJOMZZ DS CL2 message 'zz' field 02760000
      SPACE 02770000
* 02780000
* End of OS/390 DCE Application Support changes to 02790000
* default DFSLUEE0 exit routine as supplied by IMS/ESA. 02800000
* 02810000
      EJECT 02820000
      PARM_LIST DSECT 02830000
      PARM_IOFLAG DS A ADDR OF IO FLAG WORD 02840000
      PARM_MESSAGE_SEG DS A ADDR OF MESSAGE SEGMENT 02850000
      PARM_TRAN_CODE DS A ADDR OF TRAN CODE 02860000
      PARM_LUNAME DS A ADDR OF LU NAME 02870000
      PARM_USRID DS A ADDR OF USER ID 02880000

```

Figure 43 (Part 6 of 7). ASULUEE0 Exit Routine

PARM_RETURN_CODE	DS	A	ADDR OF RETURN CODE	02890000
PARM_LTERM	DS	A	ADDR OF LTERM NAME	02900000
PARM_MODNAME	DS	A	ADDR OF MOD NAME	02910000
PARM_USR_TABLE	DS	A	ADDR OF USER TABLE	02920000
PARM_MSG_FLAG	DS	A	ADDR OF MSG FLAG	02930000
*	REQUATE			02940000
R0	EQU	0		02950000
R1	EQU	1		02960000
R2	EQU	2		02970000
R3	EQU	3		02980000
R4	EQU	4		02990000
R5	EQU	5		03000000
R6	EQU	6		03010000
R7	EQU	7		03020000
R8	EQU	8		03030000
R9	EQU	9		03040000
R10	EQU	10		03050000
R11	EQU	11		03060000
R12	EQU	12		03070000
R13	EQU	13		03080000
R14	EQU	14		03090000
R15	EQU	15		03100000
	END			03110000

Figure 43 (Part 7 of 7). ASULUEE0 Exit Routine

Appendix B. Application Support Server Entry in CDS

Table 22 provides the details of the CDS entry for the Application Support server.

All object classes and attribute types that are defined for z/OS DCE have Object Identifiers that begin with **1.3.18.0.2.**

Attribute Name	Attribute Type	Required/Optional	No. of Values	Value/Syntax	Usage
CDS_Class 1.3.22.1.3.15	CDS-assigned	Required	1	1.3.18.0.2.6.1	Identifies the class of this entry.
CDS_ClassVersion 1.3.22.1.3.16	CDS-assigned	Required	1	1.0	Class version 1.0
CDS_ObjectUUID 1.3.22.1.3.17	CDS-assigned	Required	1	UUID	Reserved
Description	1.3.18.0.2.4.1	Optional	1	char	Description of a server
RPC Object UUIDs (RPC_ObjectUUIDs)	RPC-assigned	Required	N	RPC-defined	Object UUID for interfaces this server exports. The Application Support server administrative program assigns this.
RPC Binding	RPC-assigned	Required	N	RPC-defined	RPC-defined
Installed interfaces (Installed_IFs)	1.3.18.0.2.4.3	Required	N	char(8)	Table of installed interface stub module names
Subsystem information (SubSystem_Info)	1.3.18.0.2.4.4	Required	1	struct of char	Subsystem-specific attachment information. This attribute is passed to the attachment support during initialization.

Appendix C. Server States

An Application Support server that has been started is in a certain condition, called the **server state**. The server state varies, depending on the actions directed to it by the administrator or by the error conditions that the Application Support server encounters. This appendix discusses the various server states:

- Initializing State
- Quiescent State
- Starting State
- Operational State
- Stopping State

Initializing State

Once started, the Application Support server assumes this state, where it performs the following series of initialization steps to enable it to participate in DCE:

1. Logs on to DCE.
2. Builds the ID mapping table.
3. Reads the configuration parameters from the Application Support server's CDS entry.
4. Initializes the attachment to either CICS or IMS subsystems.
5. Configures the ACL Manager.
6. Creates the **protocol sequences** that the underlying communication services support.
7. **Exports** the management interfaces to the directory.
8. Registers the management interfaces with the endpoint map service.
9. Registers the management interfaces with the RPC Runtime.
10. Starts RPC Runtime executor threads to *listen* to incoming calls.

While in this state, the Application Support server cannot accept any remote procedure calls.

Quiescent State

The Application Support server enters this state by either of two ways:

- The server has completed the previously described initialization steps.
- The administrator stopped the Attachment Facility.

In this state, the following tasks can be performed on the Application Support server:

- Attach the Application Support server to the CICS or IMS subsystem.
- Install CICS or IMS transaction program interfaces.
- Uninstall CICS or IMS transaction program interfaces.
- Display Application Support server operation data.
- Display the list of all supported interfaces.
- Display data on a specific interface.

Starting State

The server enters this state from the quiescent state when either of the following occurs:

- The administrator initiates the attachment of the Application Support server to the CICS or IMS subsystem by selecting Attach Application Support Server to Subsystem from the Primary Options Panel.
- Attachment is initiated using the MODIFY operator command, or by setting the `_ASU_AUTOSTART` environment variable to YES.

In this state, the Application Support server performs the following operations that allow DCE clients to call CICS or IMS transactions.

- Starts an attachment to the CICS or IMS subsystem that has the CICS or IMS transaction programs supported by the specific instance of an Application Support server.
- Registers the CICS or IMS transaction program interfaces supported by the instance of Application Support server with the RPC Runtime.
- Restarts the proper number of RPC Runtime executor threads to *listen* to incoming calls.

While in this state, the Application Support server accepts only **Display** commands from the administrator.

Operational State

When the Application Support server has completed preparing the CICS or IMS transaction programs for RPC calls while in the starting state, the Application Support server is said to be in the operational state. While Application Support server is in this state, DCE clients can call the CICS and IMS transaction programs that the specific instance of Application Support server supports. The Application Support server can also accept requests for the following management functions while in this state:

- Detach the Application Support server from the CICS or IMS subsystem.
- Install CICS or IMS transaction program interfaces.
- Uninstall CICS or IMS transaction program interfaces.
- Display operation data about the instance of Application Support server.
- Display the list of all interfaces supported by the instance of Application Support server.
- Display data about a specific interface.
- For AS IMS with ISC only, display IMS session information.
- Reload the Identity Mapping file.

Stopping State

The Application Support server enters this state if any of the following occurs:

- An internal systems error is detected.
- The administrator has initiated the detachment of the Application Support server from the CICS or IMS subsystem by selecting Detach Application Support Server from Subsystem from the Primary Options Panel or running the MODIFY operator command.

When the Application Support server enters this state, it stops accepting RPC calls from DCE clients on behalf of the CICS or IMS transaction programs that it supports. While in this state, the following events occur:

- The Application Support server ceases to accept new requests for execution of CICS or IMS transactions programs.

- All client requests that are already in progress in the CICS or IMS address space are allowed to complete.

While in this state, the Application Support server accepts only Display commands from the administrator. Upon completion of the preceding operations, the Application Support server reverts to the quiescent state.

Appendix D. Interface States

Application Support server interfaces, which are the transaction programs offered by the Application Support server can be in a certain condition called the **interface state**, depending on the actions performed on it by the administrator. An interface can be in any of the following three states:

- Installed State
- Draining State
- Uninstalled State

This appendix explains these states.

Installed State

An interface is in the Installed state if:

- The interface name is found in the interface list associated with the server.
- The interface has been exported to the DCE directory after an install request has been processed.

Draining State

An interface is in the Draining State after an Uninstall request has been processed by the Application Support server, and while waiting for client requests made for that interface (which are already in progress) to be completed.

Uninstalled State

An interface is in the Uninstalled state if:

- The interface name is not found in the installed interface list associated with the Application Support server.
- The interface is not exported to the DCE directory.

Installed interfaces are gracefully uninstalled to allow all client requests already in progress to complete execution.

Appendix E. ASUVSMDF IDL File

The following file defines all user types used in server management interfaces. This code is in **/usr/lpp/dceas/share/include/admin/asuvsmdf.idl**.

```
/******  
/*                                                                 */  
/* OS/390 DCE Application Support                               */  
/*                                                                 */  
/* Licensed Materials - Property of IBM                       */  
/*                                                                 */  
/* 5647-A01 (C) Copyright IBM Corporation 1994, 1997        */  
/*                                                                 */  
/******  
  
/*=====*/  
/*                                                                 */  
/* Name      : asuvsmdf.idl                                   */  
/*                                                                 */  
/* Description: defines all user types used in the server    */  
/*              management i/f                               */  
/*                                                                 */  
/*=====*/  
  
interface veneertypes  
{  
  
const long max_xifs =500; /* max. number of i/fs can be installed */  
  
const long mvs_name_size = 8; /* MVS name is max of 8 characters */  
typedef [string] char mvs_name_t [mvs_name_size+1]; /* add 1 for  
                                                    null-terminated*/  
const long max_dce_name_size =1024; /* dce name is max of 1024 */  
                                           /* characters */  
typedef [string] char dce_name_t [max_dce_name_size+1]; /* plus 1 for  
                                                    null-terminated */  
  
/*-----*/  
/* i/f vector, each i/f is identified by the 8-byte */  
/* mvs name (stub module) and is null terminated */  
/*-----*/  
typedef struct {  
    long number_of_xifs;  
    [length_is(number_of_xifs)] mvs_name_t name[max_xifs];  
} xif_vector_t;  
  
/*-----*/  
/* i/f management vector : it contains a list of */  
/* interfaces and action to be performed for each */  
/* interface. */  
/*-----*/
```

Figure 44 (Part 1 of 6). ASUVSMDF IDL File

```

typedef struct {
    long number_of_xifs;
    [length_is(number_of_xifs)] char action[max_xifs]; /* U or I */
    [length_is(number_of_xifs)] mvs_name_t name[max_xifs];
} xif_mgmt_vector_t;

const char mgmt_if_uninstall='U';
const char mgmt_if_install  ='I';

/*-----*/
/* i/fs update (install/un-install) return code type*/
/* The value of the return code is defined in the */
/* constant below:                               */
/*-----*/
typedef struct {
    long number_of_xifs;
    [length_is(number_of_xifs)] unsigned32 status [max_xifs];
} xif_vector_status_t;

/*-----*/
/* i/f data to be returned to the administration */
/* client when a specific i/f to be displayed.   */
/*-----*/
typedef struct {
    unsigned16 major_version;
    unsigned16 minor_version;
    unsigned16 num_ops;
    unsigned16 if_type;
} xif_data_t; /* to be defined */

/*-----*/
/* time and date structure                        */
/* time obtained from ctime without the \n character */
/* format :                                       */
/*      Mon Jul 16 02:03:55 1987\0              */
/*-----*/

typedef [string] char date_time_t [26];

/*-----*/
/* server data to be returned to the administration */
/* client when the server data is to be displayed.  */
/*-----*/
typedef struct
{
    unsigned32 number_executors; /* executor threads created */
    unsigned32 number_xifs_installed; /* i/fs installed */
}

```

Figure 44 (Part 2 of 6). ASUVSMDF IDL File

```

dce_name_t server_princ_name;      /* server principal name */
dce_name_t who_started_stopped_attachment; /* who stopped attachment */
mvs_name_t mvs_id_start_stop_attachment; /* who stopped attachment */
date_time_t server_start_time;     /* time when server started */
date_time_t started_stopped_time;  /* time attachment started or */
                                   /* stopped : */
                                   /* started_time if state=operating */
                                   /* stopped_time if state!=operating*/
char subsystem_info [33];          /* subsystem info */

date_time_t genned;
dce_name_t server_cds_name;        /*server CDS name */

} vserver_data_t;

/*-----*/
/* session information to be returned to the */
/* administration client when session info is */
/* requested */
/*-----*/

typedef struct
{
    char   isc_device_name[8]; /* ISC session dev name */
    char   tran_code[8];      /* current tran code */
    char   tran_type;         /* current tran type */
} asuv_sess_entry_t;

/*-----*/
/* Generic management return code type. */
/* The value of the return code is defined in the */
/* constants below: */
/*-----*/
typedef struct {
    unsigned32 rc;          /* completion code of the management call */
    long int state;         /* current server state */
    long int state_reason; /* current reason code saved by the server */
    uuid_t error_corr;     /* error correlator VSC 92/03/23 */
} vmgmt_rc_t;

/* generic completion code for all server managemet calls */
const long vmgmt_rc_ok = 0x00000000; /* successful */
const long vmgmt_call_rejected = 0x10000001; /* invalid call sequence*/
const long vmgmt_authentication_failure = 0x10000002;
                                   /*e.g,call not authenticated*/

/* specific return code for start/stop attachment calls */
const long vmgmt_attachment_error = 0x10000003; /* call rejected by */
                                   /* adaptor. */

```

Figure 44 (Part 3 of 6). ASUVSMDF IDL File

```

/* specific return code for session information calls */
const long vmgmt_no_sess = 0x10000004; /* no sessions found for */
/* principal */
const long vmgmt_sess_fail = 0x10000005; /* call failed */

/* specific return code for install/uninstall/display i/f calls */
const long vmgmt_if_ok = 0x00000000; /* call successful */
const long vmgmt_if_not_found = 0x00000001; /* Xstub not found */
const long vmgmt_if_dir_error = 0x00000002; /* cannot be exported */
/* to directory */
const long vmgmt_if_epm_error = 0x00000003; /* cannot be */
/* registered with ep*/
const long vmgmt_if_rt_error = 0x00000004; /* cannot be */
/* registered with R/T*/
const long vmgmt_if_loader_error= 0x00000005; /* a loader error */
/* occurred */
const long vmgmt_if_already_inst= 0x00000006; /* i/f already */
/* installed */
const long vmgmt_if_too_many = 0x0000000E; /* too many i/fs */
/* installed */
const long vmgmt_if_not_installed=0x00000008; /* i/f is not */
/* installed */
const long vmgmt_if_inst_nt_found=0x00000009; /* installed but not */
/* in stub library */
const long vmgmt_if_uninst_in_progress=0x0000000A; /* i/f being */
/* uninstall */
const long vmgmt_if_duplicate_if =0x0000000B; /* two Xstubs have */
/* same i/f uuid and */
/* version. */

/* vsc 92/03/23 end */

/* specific return codes for reload identity mapping table calls */
const long vmgmt_cant_access_idmap_file = 0x00000001;
const long vmgmt_cant_build_hash_table = 0x00000002;
const long vmgmt_racf_selected = 0x00000003;

/* veneer server states */
const long vensrv_initializing = 0x00000001; /* server initializing*/
const long vensrv_quiescent = 0x00000002; /* attachment stopped */
const long vensrv_starting = 0x00000003; /* attachment starting*/
const long vensrv_operating = 0x00000004; /* attachment started */
const long vensrv_stopping = 0x00000005; /*attachment stopping*/
const long vensrv_terminating = 0x00000006; /* server terminating */

/*-----*/
/* Server state reason code : */
/* there are 3 types of server state reason code */
/* 1 - state reason code set by the server */
/* 2 - state reason code set by the CICS adaptor */
/* 3 - state reason code set by the IMS adaptor */
/*-----*/

```

Figure 44 (Part 4 of 6). ASUVSMDF IDL File

```

/*****/
/* type 1 state reason - set by the server with a negative value */
/*****/

const long vensrv_init          =-1; /* server just initialized */
const long vensrv_user_start   =-2; /* start attachment by user */
const long vensrv_user_stop    =-3; /* stop attachment by user */
const long vensrv_intrn_stop   =-4; /* stop due to internal      */
                                   /* server error                */
const long vensrv_epm_failure  =-5; /* stop attachment due to EPM */
                                   /* failure                      */
const long vensrv_dir_failure  =-6; /* stop attachment due to dir */
                                   /* failure                      */
const long vensrv_rt_failure   =-7; /* stop attachment due to     */
                                   /* run time failure.          */

/*****/
/* type 2 state reason - set by CICS adaptor with a positive value */
/*****/

const long vensrv_storage_allocation_err    =1;
/*
const long vensrv_initialization_failure    =2;
const long vensrv_insufficient_num_sess    =3;
*/
const long vensrv_eot_rm_not_established    =4;
const long vensrv_logic_error               =5;
const long vensrv_no_cics                   =6;
/*
const long vensrv_tcb_initialize_failure    =7;
const long vensrv_unable_to_alloc_a_pipe   =8;
const long vensrv_unknown_cics_error       =9;
*/
const long vensrv_no_irc_started            =10;
const long vensrv_DFHXXXXX_load_failed     =11;
const long vensrv_init_user_error          =12;
const long vensrv_init_user_system_error   =13;
const long vensrv_ssi_verify_failed        =14;
const long vensrv_open_pipe_system_error   =15;
const long vensrv_alloc_pipe_system_error  =16;
const long vensrv_irc_logon_failure        =17;
const long vensrv_no_pipe                  =18;
const long vensrv_irc_connect_failure      =19;
const long vensrv_allocate_pipe_user_error =20;
const long vensrv_open_pipe_user_error     =21;
const long vensrv_close_pipe_user_error    =22;
const long vensrv_irc_disconnect_failure   =23;
const long vensrv_close_pipe_system_error  =24;
const long vensrv_dealloc_pipe_user_error  =25;
const long vensrv_irc_logoff_failure       =26;

```

Figure 44 (Part 5 of 6). ASUVSMDF IDL File

```

const long vensrv_dealloc_pipe_sys_error      =27;
const long vensrv_dpl_user_error             =28;
const long vensrv_dpl_system_error           =29;
const long vensrv_invalid_version_number     =30;
const long vensrv_invalid_cics_release       =31;
const long vensrv_invalid_connection_defn    =32;
const long vensrv_unable_to_create_estae     =33;
const long vensrv_unexpected_error           =34;

/*****
/* type 3 state reason - set by IMS adaptor with a positive value. */
/* Start IMS state reasons at 100                                     */
*****/

const long vensrv_fatal_internal_error       =100;
const long vensrv_fatal_mutex_error          =101;
const long vensrv_fatal_memory_error         =102;
const long vensrv_fatal_vtam_error           =103;
const long vensrv_invalid_password           =104;
const long vensrv_no_vtam                    =105;
const long vensrv_invalid_subsystem_id       =106;
const long vensrv_log_writer_open_failed     =107; /*@B1704JS*/
}

```

Figure 44 (Part 6 of 6). ASUVSMDF IDL File

Appendix F. Administrative APIs Example Client Program

The following example shows how you might code a client application that uses the Application Support administrative APIs. This code is in `/usr/lpp/dceas/examples/common`.

```
/*-----*/
This file contains sample code. IBM provides this code on an
'as is' basis without warranty of any kind, either express
or implied, including but not limited to, the implied warranties
of merchantability or fitness for a particular purpose.

5647-A01 OS/390 DCE Application Support

Sample code of a client application to administer an Application Support
server.

Server type: AS IMS APPC or OTMA
Client program language : C

To run this program,
you must be authorized by a dce_login
usage: admin [server CDS entry name] {interface name}
-----*/

/*-----*/
/* required include statements to resolve variables */
/*-----*/
#include <stdio.h>
#include "asuvsmif.h"
#include <dce/dce_error.h>

/*-----*/
/* type definitions */
/*-----*/
unsigned32 get_binding();
unsigned32 st;
unsigned char et[dce_c_error_string_len];
int sst;
#define BINDING_LEN 256
#define STRING_LEN 80
rpc_binding_handle_t h;
uuid_t requester_uuid;
unsigned_char_t *string_binding;
unsigned_char_t protseq[STRING_LEN];
unsigned_char_t hostid[STRING_LEN];
unsigned_char_t zendpoint[STRING_LEN];
static ndr_char nil_string[] = "";
vmgmt_rc_t mrc;
xif_mgmt_vector_t ifs;
xif_vector_status_t update_rc;
xif_vector_t ifs_dup;
xif_data_t xif_data;
vserver_data_t server_data;
```

Figure 45 (Part 1 of 12). Administrative APIs Example Client Program

```

asuv_sess_entry_t IMS_session_info;
int i,j;
idl_char *princ_name_p;

/*-----*/
/* Local Function Declarations */
/*-----*/
static void InstallInterface();
static void UninstallInterface();
static void QueryInterface();
static void QueryInterfaces();
static void StartAttachment();
static void StopAttachment();
static void ReloadIDMAP();
static void DisplayServer();
static void QueryIMSSession();
static void Help();
static char GetCommand();
static void Init();
static void ProcAdmin();

/*-----*/
/* main program */
/*-----*/
main(ac,av)
int ac;
char *av[];
{
    Init(ac,av);
    strcpy((char *) (ifs.name[0]),av[2]);
    ProcAdmin(ifs.name);
}

/*-----*/
/* initialization check to see if there are inputs */
/* find the binding handle for the specified server */
/*-----*/
static void Init(ac,av)
int ac;
char *av[];
{
    printf("Running interface testcases \n");
    printf("doing rpc_binding.....\n");

    if (ac < 2)
    {
        printf("usage : %s ../AS/IMS/dev1 {MNMAXASS}\n",av[0]);
        printf("  ../AS/IMS/dev1 - server CDS entry name - MNMAXASS interface name \n");
        exit(1);
    }
}

```

Figure 45 (Part 2 of 12). Administrative APIs Example Client Program

```

get_binding(av[1]);

/*-----*/
/* returns server's principal name */
/*-----*/
rpc_mgmt_inq_server_princ_name(h,
                               rpc_c_authn_dce_secret,
                               &princ_name_p,
                               &st);

    if (st != rpc_s_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf("Cannot inquire principal name - %d\n", st);
        printf("-%s\n",et);
        exit(-1);
    }

printf("server principal name = %s\n",princ_name_p);
/*-----*/
/* setting authentication information */
/*-----*/
rpc_binding_set_auth_info(h,
                          princ_name_p,
                          rpc_c_protect_level_pkt,
                          rpc_c_authn_dce_secret,
                          NULL,
                          rpc_c_authz_dce,
                          &st);

    if (st != rpc_s_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf("Cannot set authentication info - %d\n", st);
        printf("-%s\n",et);
        exit(-1);
    }
}

/*-----*/
/* Process administration commands */
/*-----*/

static void ProcAdmin(name)
{
    char command;

    fprintf(stderr,"interface Name = %s \n",name);

    do{
        command = GetCommand(" Command: ");

        switch(command) {
            case '?':
                Help();
                break;

```

Figure 45 (Part 3 of 12). Administrative APIs Example Client Program

```

    case 'I':
        fprintf(stderr,"about to call install interface \n");
        InstallInterface(ifs.name);
        break;
    case 'U':
        fprintf(stderr,"about to call uninstall interface \n");
        UninstallInterface(ifs.name);
        break;
    case 'Q':
        fprintf(stderr,"about to call query interface \n");
        QueryInterface(ifs.name);
        break;
    case 'Y':
        fprintf(stderr,"about to call query interfaces \n");
        QueryInterfaces(ifs.name);
        break;
    case 'S':
        fprintf(stderr,"about to call start attachment \n");
        StartAttachment(ifs.name);
        break;
    case 'P':
        fprintf(stderr,"about to call stop attachment \n");
        StopAttachment(ifs.name);
        break;
    case 'R':
        fprintf(stderr,"about to call Reload Identity Mapping Table \n");
        ReloadIDMAP(ifs.name);
        break;
    case 'D':
        fprintf(stderr,"about to call Display current server data/status \n");
        DisplayServer(ifs.name);
        break;
    case 'M':
        fprintf(stderr,"about to call Query IMS Session Info \n");
        QueryIMSSession(ifs.name);
        break;
    default:
        fprintf(stderr,"\n");
        }
    } while (command != 'c' && command != 'C');
}

/*-----*/
/* GetCommand Function */
/*-----*/

static char GetCommand(prompt)
char *prompt;
{
    char command;
    fprintf(stderr," %s", prompt);
    command = getchar();
    return(toupper(command));
}

```

Figure 45 (Part 4 of 12). Administrative APIs Example Client Program

```

/* HELP function */
static void Help()
{
    fprintf(stderr,"Valid Commands are: \n");
    fprintf(stderr,"I : Install Interface \n");
    fprintf(stderr,"U : Uninstall Interface \n");
    fprintf(stderr,"Q : Query Interface \n");
    fprintf(stderr,"Y : Query Interfaces \n");
    fprintf(stderr,"S : Start Attachment \n");
    fprintf(stderr,"P : Stop Attachment \n");
    fprintf(stderr,"R: Reload IDMAP \n");
    fprintf(stderr,"D : Display Server \n");
    fprintf(stderr,"M : Query IMS session for ISC adapter \n");
    fprintf(stderr,"C : Cancel \n");
    fprintf(stderr,"? : Help \n");
}
/*-----*/
/* install interface */
/*-----*/

static void InstallInterface(name)
char *name;
{
    strcpy((char *) (ifs.name[0]),name);
    printf("install interface %s\n",ifs.name);
    ifs.action[0] = mgmt_if_install;
    ifs.number_of_xifs = 1;
    st = vensrv_mgmt_if_mgmt (h, &ifs,&ifs_dup, &update_rc, &mrc);
    if (st != error_status_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf (" an rpc error is detected , st =%d \n", st);
        printf (" -%s \n", et);
        return;
    }
    if (update_rc.status[0] != vmgmt_if_ok)
    {
        printf ("install interface call rejected %x \n",update_rc.status[0] );
        if (update_rc.status[0] == vmgmt_if_already_inst)
            printf ("Interface is installed already and you are trying to install it again \n");
    }
    else
        printf ("install interface call accepted %x \n",update_rc.status[0] );
    return;
}

/*-----*/
/* uninstall interface */
/*-----*/

static void UninstallInterface(name)
char *name;
{
    strcpy((char *) (ifs.name[0]),name);
    printf("uninstall interface %s\n",ifs.name);
}

```

Figure 45 (Part 5 of 12). Administrative APIs Example Client Program

```

    ifs.action[0] = mgmt_if_uninstall;
    ifs.number_of_xifs = 1;
    st = vensrv_mgmt_if_mgmt (h, &ifs, &ifs_dup, &update_rc, &mrc);
    if (st != error_status_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf (" an rpc error is detected , st =%d \n", st);
        printf (" -%s \n", et);
        return;
    }
    if (update_rc.status[0] == vmgmt_if_ok)
        printf ("uninstall interface call accepted %x \n", update_rc.status[0]);
    else
        printf ("uninstall interface call rejected %x \n", update_rc.status[0]);
    {
        if (update_rc.status[0] == vmgmt_if_not_installed)
            printf ("Interface is not installed and you are trying to uninstall it !!! \n");
    }
    return;
}

/*-----*/
/* start attachment */
/*-----*/

static void StartAttachment(name)
char *name;
{
    printf("Start Attachment \n");
    st = vensrv_mgmt_start_attachment (h, &mrc);
    if (st != error_status_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf (" an rpc error is detected , st =%d \n", st);
        printf (" -%s \n", et);
        return;
    }
    if (mrc.rc != vmgmt_rc_ok)
    {
        printf ("start attachment call rejected %x \n",mrc.rc);
        if (mrc.rc == vmgmt_authentication_failure)
            printf (" start attachment authentication error \n");
        if (mrc.rc == vmgmt_attachment_error)
            printf (" start attachment error \n");
        else
            printf ("start attachment call rejected error \n");
    }
    else
        printf ("start attachment call accepted %x \n",mrc.rc);
    return;
}

```

Figure 45 (Part 6 of 12). Administrative APIs Example Client Program

```

/*-----*/
/* stop attachment */
/*-----*/

static void StopAttachment(name)
char *name;
{
    printf("Stop Attachment \n");
    st = vensrv_mgmt_stop_attachment (h, &mrc);
    if (st != error_status_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf (" an rpc error is detected , st =%d \n", st);
        printf (" -%s \n", et);
        return;
    }
    if (mrc.rc != vmgmt_rc_ok)
    {
        printf ("stop attachment call rejected %x \n",mrc.rc);
        if (mrc.rc == vmgmt_authentication_failure)
            printf (" stop attachment authentication error \n");
        if (mrc.rc == vmgmt_attachment_error)
            printf (" stop attachment error \n");
        else
            printf ("stop attachment call rejected error \n");
    }
    else
        printf ("start attachment call accepted %x \n",mrc.rc);
    return;
}

/*-----*/
/* reload IDMAP */
/*-----*/

static void ReloadIDMAP(name)
char *name;
{
    printf("Reload IDMAP \n");
    st = vensrv_mgmt_reload_int (h, &mrc);
    if (st != error_status_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf (" an rpc error is detected , st =%d \n", st);
        printf (" -%s \n", et);
        return;
    }
    if (mrc.rc != vmgmt_rc_ok)
    {
        printf ("reload IDMAP call rejected %x \n",mrc.rc);
        if (mrc.rc == vmgmt_authentication_failure)
            printf (" reload IDMAP authentication error \n");
    }
}

```

Figure 45 (Part 7 of 12). Administrative APIs Example Client Program

```

        if (mrc.rc == vmgmt_cant_access_idmap_file)
            printf (" cannot access idmap file error \n");
        if (mrc.rc == vmgmt_cant_build_hash_table)
            printf(" cannot build idmap hash table \n");
        else
            printf ("reload IDMAP call rejected error \n");
    }
    else
        printf ("reload IDMAP call accepted %x \n",mrc.rc);
    return;
}

/*-----*/
/* Query interface */
/*-----*/

static void QueryInterface(name)
char *name;
{
    printf("query interface %s\n",name);
    st = vensrv_mgmt_display_if (h, name, &xif_data, &mrc);
    if (st != error_status_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf (" an rpc error is detected , st =%d \n", st);
        printf (" -%s \n", et);
        return;
    }
    if (mrc.rc == vmgmt_if_ok)
    {
        printf ("Query if call accepted: %s \n", name);
        printf ("Major version = %d \n", xif_data.major_version);
        printf ("Minor version = %d \n", xif_data.minor_version);
        printf ("Number = %d \n", xif_data.num_ops);
        printf ("Type = %d \n", xif_data.if_type);
    }
    if (mrc.rc == vmgmt_if_inst_nt_found)
    {
        printf ("Query if not found: %s \n", name);
    }
    if (mrc.rc == vmgmt_if_not_installed)
    {
        printf ("Query if not installed: %s \n", name);
    }
    if (mrc.rc == vmgmt_if_uninst_in_progress)
    {
        printf ("Query if uninstalling interface: %s \n", name);
    }
    if (mrc.rc == vmgmt_authentication_failure)
    {
        printf ("Query if auth failed: %s \n", name);
    }
    printf (" Query if: rc.state = %x\n", mrc.state);
    return;
}

```

Figure 45 (Part 8 of 12). Administrative APIs Example Client Program

```

/*-----*/
/* Query interfaces */
/*-----*/

static void QueryInterfaces(name)
char *name;

{
    printf("query interfaces\n");
    st = vensrv_mgmt_display_ifs (h,&ifs_dup, &mrc);
    if (st != error_status_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf (" an rpc error is detected , st =%d \n", st);
        printf (" -%s \n", et);
        return;
    }
    if (mrc.rc == vmgmt_rc_ok)
    {
        printf ("Query interfaces supported by the server OK: \n");
        printf ("Number of Interfaces = %d \n", ifs_dup.number_of_xifs);
        printf ("IF Name 1 = %s \n",ifs_dup.name);
        printf ("IF name 2 = %s \n",ifs_dup.name[1]);
    }
    if (mrc.rc == vmgmt_authentication_failure)
    {
        printf ("Query server interfaces auth failed\n");
    }
    if (mrc.rc == vmgmt_call_rejected)
    {
        printf ("Query server interfaces call rejected \n");
    }
    printf (" Query if: rc.state = %x\n", mrc.state);
    return;
}

/*-----*/
/* Display Server Data/Status */
/*-----*/

static void DisplayServer(name)
char *name;

{
    printf("Display Server \n");
    st = vensrv_mgmt_display_server (h, &server_data, &mrc);
    if (st != error_status_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf (" an rpc error is detected , st =%d \n", st);
        printf (" -%s \n", et);
        return;
    }
    if (mrc.rc == vmgmt_rc_ok)
    {

```

Figure 45 (Part 9 of 12). Administrative APIs Example Client Program

```

    printf ("Display Server Data/Status OK: \n");
    printf ("Number of Executors = %d \n", server_data.number_executors);
    printf ("Principal Name = %s \n", server_data.server_princ_name);
    printf ("Server Time = %s \n",server_data.server_start_time);
    printf ("Attachment Time = %s \n",server_data.started_stopped_time);
    printf ("Server CDS = %s \n",server_data.server_cds_name);
}
if (mrc.rc == vmgmt_authentication_failure)
{
    printf ("Display Server authorization failed\n");
}
if (mrc.rc == vmgmt_call_rejected)
{
    printf ("Display server call rejected \n");
}
printf (" Query if: rc.state = %x\n", mrc.state);
return;
}

static void QueryIMSSession(name)
char *name;

{
    char *act;
    printf("query IMS session\n");
    st = vensrv_mgmt_session_info (h, requester_uuid, &IMS_session_info, &mrc);
    if (st != error_status_ok)
    {
        dce_error_inq_text(st,et,&sst);
        printf (" an rpc error is detected , st =%d \n", st);
        printf (" -%s \n", et);
        return;
    }
    if (mrc.rc == vmgmt_rc_ok)
    {
        printf ("Query IMS session rpc call OK: \n");
        printf ("UUID = %d \n", requester_uuid);
    }
    if (mrc.rc == vmgmt_no_sess)
    {
        printf ("Query IMS session failed, No Session \n");
    }
    if (mrc.rc == vmgmt_call_rejected)
    {
        printf ("Query IMS session failed, Call rejected \n");
    }
    if (mrc.rc == vmgmt_sess_fail)
    {
        printf ("Query IMS session failed, Session Failure \n");
    }
    printf (" Query if: rc.state = %x\n", mrc.state);
    return;
}

```

Figure 45 (Part 10 of 12). Administrative APIs Example Client Program

```

disp_states (state)
int state;
{
    if (state ==2)
        printf (" server state = quiescent \n");
    else if (state == 3)
        printf (" server state = starting \n");
    else if (state == 4)
        printf (" server state = operational \n");
    else if (state == 5)
        printf (" server state = stopping\n");
    else if (state == 6)
        printf (" server state = terminating\n");
}

unsigned32 get_binding(name)
idl_char *name;
{
    error_status_t status;
    rpc_ns_handle_t import_context;
    unsigned_char_t *string_binding;
    int i;
    idl_char *princ_name_p;

    printf("Importing ....\n");
    status = error_status_ok;

    /* obtain a new binding vector for the security server */
    rpc_ns_binding_import_begin(rpc_c_ns_syntax_dce,
                               name,
                               NULL,
                               NULL,
                               &import_context,
                               &status);

    if (status !=rpc_s_ok)
    {
        dce_error_inq_text(status,et,&sst);
        printf("import begin failed %s\n",et);
        exit(1);
    }

    rpc_ns_binding_import_next( import_context,
                               &h,
                               &status);

    if (status !=rpc_s_ok)
    {
        dce_error_inq_text(status,et,&sst);
        printf("import next failed %s\n",et);
        exit(1);
    }
}

```

Figure 45 (Part 11 of 12). Administrative APIs Example Client Program

```
rpc_ns_binding_import_done( &import_context,
                           &status);

if (status !=rpc_s_ok)
{
    dce_error_inq_text(status,et,&sst);
    printf("import done failed %s\n",et);
    exit(1);
}
rpc_binding_to_string_binding(h,&string_binding,&st);

if (st !=error_status_ok)
{
    dce_error_inq_text(status,et,&sst);
    printf("binding to string binding failed %s\n",et);
    exit(1);
}
printf("bound to %s\n",string_binding);
return(status);

}
```

Figure 45 (Part 12 of 12). Administrative APIs Example Client Program

Appendix G. Application Support Server Messages and Codes

This appendix provides detailed explanations and recovery actions to supplement the messages and codes the Application Support server issues.

The explanations and recovery actions contained here are intended for operators, administrators, programmers, and help desk representatives who require additional information about Application Support server messages and codes.

Understanding Application Support Server Messages

This appendix contains three types of messages: error, warning, and informational. They appear in alphanumeric order in “Messages” on page 196. Each message is identified by a unique message number that is specified in the format CCCSNNNNNT, for example, the message number **ASUV00104I**. The following list describes this format:

- CCC** The first part of the message number is a three-character prefix that represents the technology that issues the message. The Application Support server technology, which the prefix ASU represents, issues all messages in this appendix. As a result, all Application Support server message numbers bear the ASU prefix.
- S** The second part of the message number is a single character that represents the specific component or subcomponent that issues the message. The Application Support server issues all the messages in this chapter. Therefore all these messages have a **V** character in this position.
- NNNN** The third part of the message number is a five-character decimal number that differentiates the message number from other message numbers that the same component or subcomponent issues.
- T** The last part of the message number is a single-character operator code that represents the type of recovery action that the operator must take in response to the message. The following characters represent the recovery actions:
- A** Indicates that the operator must take immediate action, for example, recovering from a system failure.
 - E** Indicates that the operator must take eventual action, for example, loading paper in a printer.
 - I** Indicates that the message is informational and no action is required, for example, when a command completes successfully.

These operator codes are associated with the different levels of severity, which “Severity Levels” describes.

What is First-Level Message Text?

The message number and the message text that immediately follows the message number together comprise the first-level text. This is the message information that is returned to the operator console or user display.

What is Second-Level Message Text?

The explanation and responses in “Messages” on page 196, which supplement the message number and message text, comprise the second-level text. This appendix is the only source of this second-level information.

Severity Levels

Each message contains a severity field that indicates the level of severity associated with the message. These severity levels are described in the following list:

svc_c_sev_fatal	Indicates a nonrecoverable error that probably requires manual intervention. Usually, permanent loss or damage has occurred that results in the program's ending immediately, such as database corruption. Messages of this severity have an operator code of A .
svc_c_sev_error	Indicates that an unexpected event has occurred that does not end the program and that human intervention can correct. The program continues although certain functions or services may remain unavailable. Examples include performance degradation that results in a loss of function, such as a timeout, or a specific request or action that cannot be completed, such as trying to add an object to a directory system when the object already exists. Messages of this severity have an operator code of A or E .
svc_c_sev_user	Indicates that a usage error on a public API has occurred, such as a syntax error. Messages of this severity have an operator code of A or E .
svc_c_sev_warning	Indicates one of the following: <ul style="list-style-type: none">• An error occurred that the program or system automatically corrected. An example of an error that a program corrects is when a configuration file is not found during configuration and a message is issued warning the user that certain internal defaults were used.• A condition has been detected that may be an error, depending on whether the effects of the condition are acceptable. For example, a directory is deleted and a warning message is issued that all files contained in the directory will also be deleted.• A condition exists that if left uncorrected will eventually result in an error. For example, a user receives a message while logging into DCE that warns that the user's password has expired and must be changed. Messages of this severity have an operator code of E or I .
svc_c_sev_notice	Indicates major events, such as the start of a server, completion of server initialization, or an offline server. Messages of this severity have an operator code of E or I .
svc_c_sev_notice_verbose	Indicates events of special interest, such as statistical information, key data values, use of default settings, and version information. However, it does not indicate program flow or normal events. Messages of this severity have an operator code of I .

Understanding DCE Status Codes

See *z/OS DCE Messages and Codes* for complete information about DCE Status Codes.

Understanding Message Variables

Explanations that accompany each message contain variables. The following sections explain these.

Variables Common To All Messages

All the messages share the following variables.

Variable	Definition
<i>cor-id</i>	A correlator ID is a UUID that is returned to the client as part of the <i>op_rc</i> field. It is used to locate all messages in the message log that were written as a result of a specific client error condition.
<i>function</i>	The name of the function issuing the message.
<i>file</i>	The name of the source file where the message was issued.
<i>line_number</i>	The number of the line in the source file issuing the message.

Variables Specific To Some Messages

Some of the messages use the following variables.

Variable	Definition
<i>dceid</i>	The DCE user ID of the client.
<i>error_id</i>	Contains additional error symptom information.
<i>error_information</i>	Contains a data string for use by Service personnel.
<i>error_mod</i>	Contains additional error symptom information.
<i>lstatus</i>	The local system status.
<i>mvsid</i>	The MVS user ID corresponding to the DCE user ID of the client.
<i>node</i>	The Application Support server VTAM LU name.
<i>parm1</i>	A return parameter from the VTAM macro.
<i>parm2</i>	A return parameter from the VTAM macro.
<i>reason_code</i>	The MVS reason code.
<i>return_code</i>	The return code returned to the calling client program. For further information, see the <i>z/OS DCE Application Development Reference</i> .
<i>rstatus</i>	The architecture number.
<i>sys_code</i>	The MVS system code.
<i>subpool</i>	The name of the ISC session.
<i>tcb</i>	The TCB address under which this message was issued.
<i>error_text</i>	The system error character string.

Messages

The messages are listed by message number.

ASUV00100A Application Support server cannot process Attach Facility request.

Severity: svc_c_sev_error

Explanation: The Application Support server detects an error when attempting to start or stop the Attach Facility.

System Action: The program continues.

Administrator Response: Use the Display Operational Data option of the Application Support administration program to display the state of the Application Support server. Refer to the message displayed on the panel to determine how to correct the problem.

ASUV00101A Client program is not authorized to access Application Support server.

Severity: svc_c_sev_error

Explanation: The client does not have sufficient authority, so the Application Support server rejects the client request. In most cases, the reasons for the failure are:

- Client is not logged into DCE.
- Application Support server does not have an identity mapping for the client.
- Client does not have permission to make the administrative request.

System Action: Request is not processed.

Administrator Response: Do one of the following, then try the request again:

- Log in to DCE
- Create an identity mapping for the client and start the Application Support server again or reload the identity mapping file.
- Modify the access control list (ACL) to give the client sufficient authority.

User Response: Do one of the following, then try the request again:

- Log in to DCE
 - Contact the Application Support server administrator.
-

ASUV00104I Application Support server accepts Attach Facility Start request.

Severity: svc_c_sev_notice

Explanation: The Application Support server begins to process the Attach Facility Start request. When the Application Support server completes the Attach Facility request, the **server state** is changed from **Starting** to **Operational**. The Application Support administration program Display Operational Data function can be used to determine the status of the Attach Facility Start request.

System Action: The program continues. The Start request is processed asynchronously by the Application Support server.

ASUV00105I Application Support server accepts Attach Facility Stop request.

Severity: svc_c_sev_notice

Explanation: The Application Support server begins to process the Attach Facility Stop request. When the Application Support server completes the Detach Facility request, the **server state** is changed from **Operational** to **Quiescent**. The Application Support administration program Display Operational Data function can be used to determine the status of the Attach Facility Stop request.

System Action: The program continues. The Stop request is processed asynchronously by the Application Support server.

ASUV00107A A Server Name must be specified.

Severity: svc_c_sev_error

Explanation: The administrator pressed the Enter key from an Application Support administration program panel without specifying a server name.

System Action: The program continues.

Administrator Response: Specify the CDS full name of an Application Support server in the **Server Name** field and try the request again.

ASUV00108A Option specified is not valid. Choose 1 to 7.

Severity: svc_c_sev_error

Explanation: Options 1 through 7 are valid. Either an option outside this range, or no option is specified.

System Action: The program continues.

Administrator Response: Select a valid option and try the request again.

ASUV00110A Option 2 requires the specification of a 1 to 8 character interface name.

Severity: svc_c_sev_error

Explanation: The Display Specific Interface option of the Application Support administration program requires a 1 to 8 character interface name but the **Interface Name** field is blank.

System Action: The program continues.

Administrator Response: Specify an interface name in the **Interface Name** field and try again. The **interface name** is the name of an interface load module found in the Application Support server stub library.

ASUV00111A Option specified is not valid. Choose 1 to 4.

Severity: svc_c_sev_error

Explanation: Options 1 through 4 are valid. Either an option outside of this range, or no option is specified.

System Action: The program continues.

Administrator Response: Select a valid option and try the request again, or press the PF3 key to exit from the panel.

ASUV00112I No installed interface is available for display.

Severity: svc_c_sev_notice

Explanation: There are no interfaces installed in the Application Support server.

System Action: The program continues.

ASUV00113A Application Support administration program detects status from RPC runtime that is not expected. DCE status code : *status_code*.

Severity: svc_c_sev_error

Explanation: During the administrative request, an attempt is made to communicate with the Application Support server through the RPC run time and a result that is not expected is returned. The result of the administrative request is not known.

System Action: The program continues.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912.

ASUV00114A Application Support server is not in correct state to perform operation.

Severity: svc_c_sev_error

Explanation: The Application Support server is in a state that does not allow the requested operation to be performed. The Application Support server rejects the request. The most likely reasons for failure are:

- Attempt to Start the Attach Facility when the Application Support server state is **Operational**.
- Attempt to Stop the Attach Facility when the Application Support server state is **Quiescent**.

System Action: The program continues.

Administrator Response: Use the Display Operational Data option of the Application Support administration program to display the state of the Application Support server. Refer to the *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903, appendix to determine the compatibility between administrative requests and the states of the Application Support server. Try the request again. If the problem continues, contact the service representative.

ASUV00115A Server name is required.

Severity: svc_c_sev_error

Explanation: Application Support administration program cannot process the request because the Application Support server name is not specified.

System Action: The program continues.

Administrator Response: Specify the name of an Application Support server and try the request again.

ASUV00116A CICS generic Application ID not specified but is required.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot process a request because the required CICS generic Application ID (APPLID) is not specified.

System Action: The program continues.

Administrator Response: Specify the CICS generic APPLID and try the request again, or press the PF3 key to exit from the current panel.

ASUV00117A Server generic Application ID not specified but is required.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot process a request because the required Server generic Application ID (APPLID) is not specified.

System Action: The program continues.

Administrator Response: Specify the Server generic APPLID and try the request again, or press the PF3 key to exit from the current panel.

ASUV00118I Application Support server definition is created for server *server*.

Severity: svc_c_sev_notice

Explanation: The Application Support administration program creates the definition for Application Support server *server* in the DCE Directory service.

System Action: The program continues.

ASUV00120I Application Support server definition is modified for server *server*.

Severity: svc_c_sev_notice

Explanation: The Application Support administration program modifies the definition for Application Support server *server* in the DCE Directory service. The next time the Application Support server is started, it uses the modified information.

System Action: The program continues.

ASUV00122I Application Support server definition is deleted for server *server*.

Severity: svc_c_sev_notice

Explanation: The Application Support administration program deletes the definition for Application Support server *server*. The Application Support server definition is removed from the DCE Directory service. The administrator must create a new definition for the Application Support server before it can be started again.

System Action: The program continues. If the Application Support server *server* is currently running, it continues with reduced function.

ASUV00125I Application Support server state is *state*.

Severity: svc_c_sev_notice

Explanation: The Application Support server state determines which operations can be performed. Refer to *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903, appendix to determine the compatibility between administrative requests and server states.

System Action: The program continues.

ASUV00128I Application Support server Attach Facility is not operational.

Severity: svc_c_sev_notice

Explanation: The Attach Facility has not been started. The Application Support server does not accept any transaction requests until the Attach Facility is started.

System Action: The program continues.

ASUV00129I Start Attach Facility command is issued.

Severity: svc_c_sev_notice

Explanation: The Application Support server accepts an Attach Facility Start request. If the Application Support server state is **Starting**, the Attach Facility Start request is in progress. The state changes to **Operational** once the Start Attach Facility request successfully completes.

System Action: The program continues.

ASUV00130I Stop Attach Facility command is issued.

Severity: svc_c_sev_notice

Explanation: The Application Support server accepts an Attach Facility Stop request. If the Application Support server state is **Stopping**, the Attach Facility Stop request is in progress. The state changes to **Quiescent** once the Stop Attach Facility request successfully completes.

System Action: The program continues.

ASUV00131A Application Support server Attach Facility stops because of internal error.

Severity: svc_c_sev_error

Explanation: Additional messages and diagnostic information may be recorded by the Application Support server.

System Action: The program continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00132A Application Support CICS cannot allocate storage.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility is not able to allocate storage. This should not occur during normal processing.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Increase the region size of the Application Support server and try the operation again. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00134A Application Support CICS detects condition that is not expected.

Severity: svc_c_sev_error

Explanation: An error occurs during Application Support server execution that is not expected.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00137A Application Support CICS cannot attach to CICS subsystem.

Severity: svc_c_sev_error

Explanation: In most cases, the reasons for the failure are:

- The RDO-defined connection to the CICS system from the server is out of service.
- The target CICS system is not operational.
- The CICS system is running without the IRC enabled.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Ensure that the CICS system associated with the Application Support server is operational and is logged on to the IRC. Then try the request again. If the problem continues, save the dump and contact the service representative.

ASUV00138A Application Support administration program cannot obtain DCE principal name of Application Support server.

Severity: svc_c_sev_error

Explanation: The administration program requires the Application Support server DCE principal name to authenticate its bindings to the Application Support server. The status returned from the DCE API **rpc_mgmt_inq_server_princ_name** is displayed in the following message. In most cases, the reasons for the failure are:

- The Application Support server name specified in the Server Name field is not correct.
- The Application Support server is not running.
- The Application Support server initialization is not complete.
- An RPC communications failure occurs.

System Action: The program continues with reduced function.

Operator Response: Start the Application Support server if it is not running.

Administrator Response: Depending on the information provided in the DCE status code text, do one of the following and try the request again:

- Ensure that the Server Name field contains the correct Application Support server name.
- Wait for the initialization of the Application Support server to complete.

If the problem continues, contact the service representative.

ASUV00139A Application Support administration program cannot authenticate binding handle for Application Support server.

Severity: svc_c_sev_error

Explanation: The Application Support administration program must obtain the binding handle for the Application Support server so that it can make authenticated remote procedure calls. The DCE status code text returned from the DCE API **rpc_binding_set_auth_info** is displayed in the following message.

System Action: The program continues with reduced function.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00141A Application Support administration program detects state reason that is not known. State Reason is *reason_code*.

Severity: svc_c_sev_error

Explanation: Application Support server returns a state reason to the Application Support administration program that is not known.

System Action: The program continues.

Administrator Response: Try the request again. If the problem continues, contact the service representative.

ASUV00142A Application Support administration program cannot resolve binding to the Application Support server.

Severity: svc_c_sev_error

Explanation: The Application Support administration program attempts to resolve its binding to the Application Support server but the DCE API **rpc_ep_resolve_binding** fails. The DCE status code is displayed in the following message. The Application Support administration program cannot make a request to the Application Support server. In most cases, the reason for the failure is: that the Application Support server initialization is not complete.

System Action: The program continues with reduced function.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00144A Application Support administration program cannot create UUID for Application Support server definition.

Severity: svc_c_sev_error

Explanation: The status code returned from the DCE API **uuid_create** is displayed in message ASUV00147A.

System Action: The program continues.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00145A Application Support administration program cannot create Application Support server definition.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot create a DCE directory object for Application Support server. The DCE status code text is displayed in the following message.

System Action: The program continues.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00146A Application Support administration program cannot add Application Support server subsystem information.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot add the subsystem information to the DCE directory object for the Application Support server. The DCE status code text is displayed in the following message. If this error occurs while using the Define Server Definition option of the Application Support administration program, the DCE directory object is deleted.

System Action: The program continues.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00147A DCE API uuid_create fails. Status code is *status_code*.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot create a UUID for the Application Support server. The DCE API **uuid_create** returns DCE status code *status_code*.

System Action: The program continues.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again.

ASUV00149A Application Support administration program cannot initialize context handle for DCE Directory service operations.

Severity: svc_c_sev_error

Explanation: The Application Support administration program requires a context handle to perform DCE directory service operations. The Application Support administration program cannot process the Application Support server operation. This error should not occur during normal processing.

System Action: The program continues.

System Programmer Response: Increase the region size of the Application Support administration program and try the operation again. If the problem continues, contact the service representative.

ASUV00150A Application Support administration program cannot obtain Application Support server definition in DCE directory.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot obtain the DCE directory object for the Application Support server. The following message contains the DCE status code text.

System Action: The program continues.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00151A Application Support server definition is not valid.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot perform the definition operation because the Application Support server definition is not valid. In most cases, the reasons for the failure are:

- The Application Support server name specified in the Server Name field is not correct.
- The DCE directory may contain another object with the same name, which is not an Application Support server.

System Action: The program continues.

Administrator Response: Do one of the following, then try the request again:

- Specify the CDS full name for the Application Support server.
- Use the CDS control program to determine if a duplicate server name exists in the DCE directory. If a duplicate name exists, contact the DCE administrator.
- Use the CDS control program to delete the CDS object. The Application Support administration program can be used to recreate the definition. Refer to the *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903, for more information.

If the problem continues, contact the service representative.

ASUV00152A Application Support administration program cannot delete Application Support server.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot delete the Application Support server from the DCE directory. The following message contains the DCE status code text returned from the DCE directory delete operation. In most cases, the reasons for the failure are:

- The administrator is not logged into DCE.
- The administrator does not have the authority to delete the Application Support server definition from the DCE directory.

System Action: The program continues.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again.

ASUV00153A Application Support administration program cannot create lookup context required to bind to Application Support server.

Severity: svc_c_sev_error

Explanation: The administration program requires a lookup context to bind to the Application Support server. Without this lookup context, the client program cannot communicate with the Application Support server. The following message contains the DCE status code text returned from the DCE API `rpc_ns_binding_lookup_begin`. In most cases, the reasons for the failure are:

- The Application Support server name is not valid.
- The Application Support server is not running or its initialization is not complete.

System Action: The program continues.

Operator Response: Start the Application Support server if it is not running.

Administrator Response: Do one of the following, then try the request again:

- Specify the CDS full name for the Application Support server.
- Wait for the Application Support server initialization to complete.
- Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912.

Refer to *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903, for further information. If the problem continues, contact the service representative.

ASUV00155A Application Support administration program cannot obtain binding handle for Application Support server.

Severity: svc_c_sev_error

Explanation: The administration program requires a binding handle to communicate with the Application Support server. The following message contains the DCE status code text returned from the DCE API `rpc_ns_binding_lookup_next`. In most cases, the reasons for the failure are:

- The administrator is not logged into DCE.
- The administrator does not have authority to read the DCE directory object for the Application Support server.
- The Application Support server name is not correct.
- The Application Support server does not exist in the DCE directory.
- The Application Support server is not running or its initialization is not complete.

System Action: The program continues.

Operator Response: Start the Application Support server if it is not running.

Administrator Response: Do one of the following, then try the request again:

- Log in to DCE.
- Ask the DCE administrator to give you the necessary authority to the Application Support server in the DCE directory.
- Specify the CDS full name for the Application Support server.
- Create an Application Support server definition using the Create Server Definition option of the Application Support administration program. Refer to *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903, for more information.
- Wait for the Application Support server initialization to complete.
- Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912.

If the problem continues, contact the service representative.

ASUV00156A Cannot bind to Application Support server because no binding handles exist.

Severity: svc_c_sev_error

Explanation: The administration program cannot communicate with the Application Support server because no binding handles to that server are found.

System Action: The program continues.

Operator Response: Start the Application Support server if it is not running.

Administrator Response: Ensure that the Application Support server is running and its initialization is complete. Try the request again. If the problem continues, contact the service representative.

ASUV00157A IMS LU Name not specified but is required.

Severity: svc_c_sev_error

Explanation: The IMS LU Name field is blank but is required for the Application Support administration program option.

System Action: The program continues.

Administrator Response: Specify the IMS LU Name for the Application Support server and try the request again, or press PF3 to exit from the panel.

ASUV00158A Session Logon Mode Name is not specified but is required.

Severity: svc_c_sev_error

Explanation: The Session Logon Mode Name is required for the Application Support administration program.

System Action: The program continues.

Administrator Response: Specify the Session Logon Mode Name for the Application Support server and try the request again, or press PF3 to exit from the panel.

ASUV00159A Application Support server LU Name is not specified but is required.

Severity: svc_c_sev_error

Explanation: The Application Support server LU Name is required for the Application Support administration program.

System Action: The program continues.

Administrator Response: Specify the LU Name for the Application Support server and try the request again, or press the PF3 key to exit from the panel.

ASUV00160A Application Support server device prefix not specified but is required.

Severity: svc_c_sev_error

Explanation: The Application Support server device prefix is required for the Application Support administration program.

System Action: The program continues.

Administrator Response: Specify the Application Support server Device Prefix and try the request again, or press the PF3 key to exit from the panel.

ASUV00162A Application Support server return_code *return_code* is not expected.

Severity: svc_c_sev_error

Explanation: The Application Support server returns a return code *return_code* that is not expected by the administration program.

System Action: The program continues.

Administrator Response: Try the request again. If the problem continues, contact the service representative.

ASUV00164A Application Support IMS Attach Facility detects fatal error.

Severity: svc_c_sev_error

Explanation: The Application Support IMS Attach Facility detects an error from which it cannot recover. The Application Support IMS logs additional messages, and may generate a probe and a dump. This message appears on the Display Operational Data panel of the Application Support administration program. Additional information about the error is logged by the Application Support server.

System Action: The Application Support IMS Attach Facility is stopped.

Operator Response: Save any dump and probe information collected and contact the service representative.

Administrator Response: If the problem continues, save the dump and contact the service representative.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00165A Application Support IMS Attach Facility cannot create, lock, or unlock mutex.

Severity: svc_c_sev_error

Explanation: The Application Support IMS Attach Facility encounters an error while performing a mutex operation. The Application Support IMS logs additional diagnostic information, and stops the Attach Facility. This message appears on the Display Operational Data panel of the Application Support administration program.

System Action: The program continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00166A Application Support IMS Attach Facility cannot allocate storage.

Severity: svc_c_sev_error

Explanation: The Application Support IMS Attach Facility cannot allocate storage. The Application Support IMS logs additional diagnostic data, and stops the Attach Facility. This message appears on the Display Operational Data panel of the Application Support administration program.

System Action: The Application Support IMS Attach Facility is stopped.

System Programmer Response: Increase the region size for the Application Support server and try the request again. If the problem continues, contact the service representative.

ASUV00167A Application Support IMS Attach Facility cannot open or close VTAM ACB.

Severity: svc_c_sev_error

Explanation: The Application Support IMS Attach Facility detects an error opening or closing the VTAM Access Method Control Block (ACB). This condition occurs during an Attach Facility Start request or an Attach Facility Stop request. The Application Support IMS logs additional diagnostic data. This message appears on the Display Operational Data panel of the Application Support administration program.

System Action: The Application Support IMS Attach Facility is stopped.

System Programmer Response: See the console log for more information, including a VTAM return code that explains why the open or close ACB macro fails. Then try the request again. If the problem persists, contact the service representative.

ASUV00168A VTAM ACB password is not correct or is missing.

Severity: svc_c_sev_error

Explanation: The Application Support IMS Attach Facility cannot complete its initialization or process a start attachment request because the VTAM Access Method Control Block (ACB) password is either not correct or is missing.

System Action: The program continues with reduced function.

Administrator Response: Ensure that the VTAM ACB password is correct, then try the request again.

ASUV00169A VTAM is not operational.

Severity: svc_c_sev_error

Explanation: The Application Support IMS Attach Facility cannot attach to IMS because VTAM is either not installed or is not operational. The Application Support IMS server may log additional information. This message appears on the Display Operational Data panel of the Application Support administration program.

System Action: The program continues with reduced function.

Operator Response: Start VTAM again. Try the request again.

System Programmer Response: Install VTAM if it is not already on the system.

ASUV00170A Application Support server does not support requested operation.

Severity: svc_c_sev_error

Explanation: The Display IMS Session Information option is only supported by an Application Support IMS server, but the Application Support server specified in the Server Name field is an Application Support CICS server.

System Action: The program continues.

Administrator Response: Specify an Application Support IMS server and try the request again.

ASUV00171A Application Support CICS detects IRC is not started.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot initialize because no CICS system has used IRC services since MVS was started. The Attach Facility is not initialized.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server ends.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Ensure that a CICS system in the MVS image is logged on to the IRC and try to start the Attach Facility again. The IRC can be enabled by starting a CICS system with IRCSTRT=YES specified in the SIT or by using the CEMT transaction to open IRC. If the problem continues, save the dump and contact the service representative.

ASUV00172A Application Support CICS cannot load CICS module *module_name*.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot load the module *module_name*. This module is required for the CICS system attachment. In most cases, the reason for the failure is: that CICS is not configured correctly for the Application Support server.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Ensure that the load module *module_name* resides in a library available to the Application Support server and start the Attach Facility again. Refer to the *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903, for more details. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00173A Application Support CICS detects USER_ERROR from EXCI interface Initialize_User.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility detects a USER_ERROR response from the External CICS interface (EXCI) Initialize_User.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the CICS information and dump to determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00174A Application Support CICS detects SYSTEM_ERROR from EXCI interface Initialize_User.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility detects a SYSTEM_ERROR response from the external CICS interface (EXCI) Initialize_User.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the CICS information and dump to determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00175A Application Support CICS cannot VERIFY call to MVS subsystem interface.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot perform the request because the external CICS interface (EXCI) call Initialize_User fails. The reason code indicates that the CICS EXCI could not obtain the current CICS SVC number.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Ensure that CICS is defined correctly as an MVS subsystem and try the request again. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00176A Application Support CICS detects SYSTEM_ERROR in EXCI interface Open_Pipe.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot open a pipe from the Application Support server to the target CICS system. The call to the external CICS interface (EXCI) Open_Pipe fails.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the CICS information and dump to determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00177A Application Support CICS cannot allocate pipe.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot allocate a pipe from the Application Support server to the target CICS system.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the CICS information and dump to determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00178A Application Support CICS detects failure of IRC logon.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility request to log on to IRC during allocate pipe processing fails.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Locate the error codes in the IRC data area (Interregion Control Blocks). The control block name is DFHIRSPS. Refer to the *CICS Data Areas*, LY33-6096, and take the suggested corrective action. If the problem continues, save the dump and contact the service representative. Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00179A Application Support CICS cannot open pipe to target CICS subsystem.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot find an available pipe for communication to the target CICS system. In most cases, the reason for the failure is: the RECEIVECOUNT value specified in the RDO SESSIONS definition is too low to support the number of concurrent RPC requests being received.

If this during occurs during an Attach Facility Start request, then the Application Support CICS Attach Facility writes a dump and stops the Attach Facility. Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The program continues.

System Programmer Response: Ensure that the RECEIVECOUNT value specified on the RDO SESSIONS definition for the server is greater than the value specified for the `_ASU_CICS_CONNECTIONS` environment variable. Then try the request again. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00180A Application Support CICS detects failure of IRC connect.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility detects that an IRC connect request fails during open pipe processing.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Locate the error codes in the IRC data area (Interregion Control Blocks). The control block name is DFHIRSPS. Refer to the *CICS Data Areas*, LY33-6096, and take the suggested corrective action. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00181A Application Support CICS detects failure from EXCI interface Allocate_Pipe.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot allocate a pipe from the Application Support server to the target CICS system. The call to the external CICS interface (EXCI) Allocate_Pipe fails.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the CICS information and dump to determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00182A Application Support CICS detects USER_ERROR from EXCI Interface Open_Pipe.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot open a pipe from the Application Support server to the target CICS system. The call to the external CICS interface (EXCI) Open_Pipe fails.

In most cases, the reason for the failure is: the attached CICS system ends abnormally while client RPC requests are being processed by the Application Support server.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: If the attached CICS system ends abnormally while the Attach Facility is operational, start the CICS system and start the Application Support server Attach Facility. Then try the request again. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00183A Application Support CICS detects USER_ERROR from EXCI interface Close_Pipe.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot close a pipe from the Application Support server to the target CICS system. The call to the external CICS interface (EXCI) Close_Pipe fails.

In most cases, the reason for the failure is: that the attached CICS system ends abnormally while client RPC requests are being processed by the Application Support server.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: If the attached CICS system ends abnormally while the Attach Facility is operational, start the CICS system and start the Application Support server Attach Facility. Then try the request again. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00184A Application Support CICS cannot disconnect IRC.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility detects that an IRC disconnect request fails during close pipe processing.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Locate the error codes in the IRC data area (Interregion Control Blocks). The control block name is DFHIRSPS. Refer to the *CICS Data Areas*, LY33-6096, and take the suggested corrective action. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00185A Application Support CICS detects SYSTEM_ERROR from EXCI interface Close_Pipe.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot close a pipe from the Application Support server to the target CICS system. The call to the external CICS interface (EXCI) Close_Pipe fails.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the CICS information and dump to determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00186A Application Support CICS detects USER_ERROR from EXCI interface Deallocate_Pipe.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot deallocate a pipe from the Application Support server to the target CICS system. The call to the external CICS interface (EXCI) Deallocate_pipe fails.

In most cases, the reason for the failure is: that the attached CICS system ends abnormally while the Attach Facility is operational.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: If the attached CICS system ends abnormally while the Attach Facility is operational, start the CICS system and start the Application Support server Attach Facility. Then try the request again. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00187A Application Support CICS detects IRC logoff system error.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility is trying to log off the IRC during deallocate pipe processing but the request fails.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Locate the error codes in the IRC data area (Interregion Control Blocks). The control block name is DFHIRSPS. Refer to the *CICS Data Areas*, LY33-6096, and take the suggested corrective action. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00188A Application Support CICS detects SYSTEM_ERROR from EXCI interface Deallocate_Pipe.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot deallocate a pipe from the Application Support server to the target CICS system.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the CICS information and dump to determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00189A Application Support CICS detects USER_ERROR from EXCI interface DPL_Request.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot communicate a request from the Application Support server to the target CICS system.

In most cases, the reason for the failure is: that the attached CICS system ends abnormally while client RPC requests are being processed in the Application Support server.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: If the attached CICS system ends abnormally while the Attach Facility is operational, start the CICS system and start the Application Support server Attach Facility. Then try the request again. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00190A Application Support CICS detects SYSTEM_ERROR from EXCI interface DPL_Request.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot communicate a request from the Application Support server to the target CICS system.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the CICS information and dump to determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00191A Application Support CICS detects target CICS subsystem does not support external CICS interface.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot communicate a request from the Application Support server to the target CICS system. In most cases, the reasons for the failure are:

- The external CICS interface is not installed.
- The external CICS interface is not supported by the attached CICS system.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Ensure that the external CICS interface is installed correctly on the target CICS system. Then try the request again. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00192A Application Support CICS detects CONNECTION definition in CICS is not valid.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility cannot communicate a request from the Application Support server to the target CICS system. In most cases, the reason for the failure is: the CONNECTION definition does not specify an external CICS interface (EXCI) protocol.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Do one of the following, then try the request again:

1. Ensure that the CONNECTION definition for the Application Support server specifies an EXCI protocol.
2. Close the IRC.
3. Install the connection definition again.
4. Open the IRC again.
5. Start the Attach Facility.

6. Try the request again.

If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00193A RPC time out occurs.

Severity: svc_c_sev_error

Explanation: The RPC runtime detects that an RPC time out occurs while waiting for the Application Support server to respond to a RPC call. RPC will time out if the Application Support server does not respond for some specified time. In most cases, the reasons for the failure are:

- The Application Support server is stopped.
- The Application Support server is not listening for requests. The Application Support server does not listen for requests while initializing, starting or stopping the Attach Facility, or terminating.
- The system is heavily loaded.
- The RPC time out value is too low.

System Action: The program continues.

Operator Response: Start the Application Support server if it is not running.

Administrator Response: Ensure that the Application Support server is running and its initialization is complete.

Ensure that the RPC time out value is appropriate for your environment. The default value is 30 seconds. You can increase the value by changing the time out environment variable.

Try the request again. If the problem continues, contact the service representative.

ASUV00194A RPC communication failure occurs.

Severity: svc_c_sev_error

Explanation: The RPC run time is not communicating with the Application Support server and returns from the RPC call. In most cases, the reasons for the failure are:

- The Application Support server is stopped.
- The Application Support server is not listening for requests. The Application Support server does not listen for requests while initializing, starting or stopping the Attach Facility, or when terminating.
- The system is heavily loaded.

System Action: The program continues.

Operator Response: Start the Application Support server, if it is not running.

Administrator Response: Ensure that the Application Support server is running and its initialization is complete. Try the request again. If the problem continues, contact the service representative.

ASUV00195A Cannot register interfaces with RPC endpoint mapper.

Severity: svc_c_sev_error

Explanation: During Application Support Attach Facility start, the Application Support server cannot register its interfaces with the RPC endpoint mapper. The Application Support server must register the interfaces to make them available to client programs. The Application Support server stops the Attach Facility. Message ASUV00573A contains additional information about the error.

System Action: The program continues with reduced function.

System Programmer Response: Refer to ASUV00573A for additional information.

ASUV00196A Attach Facility stops because of directory failure.

Severity: svc_c_sev_error

Explanation: Additional messages may be generated by the DCE Directory service.

System Action: The program continues.

System Programmer Response: Refer to the console log for additional messages from the DCE directory service. Start the Attach Facility and try the request again. If the problem continues, contact the service representative.

ASUV00197A Cannot register interfaces with RPC run time.

Severity: svc_c_sev_error

Explanation: The Application Support server cannot start the Attach Facility because it cannot register its interfaces with the RPC run time. The Application Support server logs additional diagnostic information in the console log.

System Action: The program continues.

Administrator Response: Correct the error based on the messages logged in the console log. Start the Attach Facility and try again. If the problem continues, contact the service representative.

ASUV00198A Application Support server operational data cannot be displayed because subsystem identifier is not valid.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot process the Display Operational Data request because the subsystem information returned by the Application Support server is not valid. The Application Support administration program can only display operational data for Application Support CICS and Application Support IMS servers. The subsystem identifier returned does not match either the Application Support CICS identifier or the Application Support IMS identifier.

System Action: The program continues.

Administrator Response: Refer to the Application Support server message log for further information about the error.

ASUV00199A Application Support administration program cannot obtain login context.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot obtain the login context using the `sec_login_get_current_context` API. In most cases, the reason for the failure is: that the user did not log in.

System Action: The program continues with reduced function.

Administrator Response: Log in to DCE and try the request again.

ASUV00225A Application Support administration program cannot validate or certify identity. DCE Login fails.

Severity: svc_c_sev_error

Explanation: The administrator uses the Perform DCE Login option of the Application Support administration program, but the login is not successful. The following message contains the DCE status code text. In most cases, the reasons for the failure are:

- The principal name does not exist.
- The principal account is not valid.
- The password is not correct.
- The Security daemon is not available.

System Action: The program continues with reduced function.

Administrator Response: Do one of the following, then try the request again:

- Specify an existing principal name.
- Enter the correct password.
- Use the DCE status code text to correct the error.

If the problem continues, contact the service representative.

ASUV00226I DCE security service warns that password must be changed.

Severity: svc_c_sev_error

Explanation: The DCE login is successful, but the password used to perform a DCE login has remained constant for a longer period of time than is recommended by standard security practices.

System Action: The program continues.

Administrator Response: Use the DCE administrative interface (**dcecp**) to change the password.

ASUV00227A DCE Security service cannot obtain network credentials. Login fails.

Severity: svc_c_sev_error

Explanation: The DCE login context must be authorized by the network, but no network credentials are available. The DCE login fails.

System Action: The program continues with reduced function.

Administrator Response: Try the request again. If the problem continues, contact the service representative.

ASUV00229A Application Support administration program cannot create network credentials for login context. Login fails.

Severity: svc_c_sev_error

Explanation: The DCE login fails because the Application Support administration program cannot create network credentials for the login context.

System Action: The program continues with reduced function.

Administrator Response: Try the request again. If the problem continues, contact the service representative.

ASUV00231A Application Support administration program cannot setup login identity. Login fails.

Severity: svc_c_sev_error

Explanation: The DCE login fails because the Application Support administration program cannot setup a login identity for the principal name. The following message contains the DCE status code text returned from the DCE API **sec_login_setup_identity**. In most cases, the reasons for the failure are:

- The principal does not exist.
- The Security daemon is not available.

System Action: The program continues with reduced function.

Administrator Response: Do one of the following, then try the request again:

- Specify a principal that exists.
- Use the DCE status code text to correct the error.

If the problem continues, contact the service representative.

ASUV00234I DCE Login is successful.

Severity: svc_c_sev_notice

Explanation: The administrator is successfully logged onto DCE as the DCE principal.

System Action: The program continues.

ASUV00235A ISPF cannot display panel *panel*. Return code=*status_code*, line *line*, file *file*.

Severity: svc_c_sev_error

Explanation: The ISPF panel *panel* could not be displayed. The ISPF panel file may be damaged, destroyed or misplaced. The ISPF return code is *status_code*. The error occurs in file *file* at line *line*.

System Action: The program continues with reduced function.

System Programmer Response: Use the panel name *panel* and the return code *status_code* to correct the error. Try the request again. If the problem continues, contact the service representative.

ASUV00236A ISPF function *function* fails. Return code=*status_code*, line *line*, file *file*.

Severity: svc_c_sev_error

Explanation: The ISPF function *function* fails with return code *status_code*. The error occurs in file *file* at line *line*.

System Action: The program continues with reduced function.

Administrator Response: Correct the error by using the ISPF return code response information found in *z/OS ISPF Dialog Developer's Guide and Reference*, SC34-4821, and *z/OS ISPF Reference Summary*, SC34-4816.

System Programmer Response: Use the ISPF function name and ISPF return code to correct the error. Try the request again. If the problem continues, contact the service representative.

ASUV00237A ISPF cannot define variable *variable*. Return code=*status_code*, line *line*, file *file*.

Severity: svc_c_sev_error

Explanation: The ISPF function DEFINE cannot define the variable *variable* that provides the link between panel fields and program variables. The DEFINE function fails with return code *status_code*. The error occurs in file *file* at line *line*.

System Action: The program continues with reduced function.

Administrator Response: Correct the error by using the ISPF return code response information found in *z/OS ISPF Dialog Developer's Guide and Reference*, SC34-4821, and *z/OS ISPF Reference Summary*, SC34-4816.

System Programmer Response: Use the ISPF return code *status_code* to correct the error. Try the request again. If the problem continues, contact the service representative.

ASUV00238A ISPF cannot obtain variable *variable*. Return code=*status_code*, line *line*, file *file*.

Severity: svc_c_sev_error

Explanation: ISPF cannot obtain the contents of a panels input field. The return code is *status_code*.

System Action: The program continues with reduced function.

Administrator Response: Correct the error by using the ISPF return code response information found in *z/OS ISPF Dialog Developer's Guide and Reference*, SC34-4821, and *z/OS ISPF Reference Summary*, SC34-4816.

System Programmer Response: Use the ISPF return code *status_code* to correct the error. Try the request again. If the problem continues, contact the service representative.

ASUV00240A Application Support administration program cannot bind to security registry site.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot process the request because it cannot bind to a security registry site. In most cases, the reasons for the failure are:

- The administrator is not logged onto DCE.
- The Security daemon is not available.

The following message contains the DCE status code or DCE status code text returned from the DCE API **sec_rgy_site_open**.

System Action: The program continues with reduced function.

Administrator Response: Do one of the following, then try the request again:

- Log in to DCE.
- Ensure that the Security daemon is available.

If the problem continues, contact the service representative.

ASUV00241A Application Support administration program cannot obtain UUID for principal name.

Severity: svc_c_sev_error

Explanation: The Application Support administration program cannot obtain the UUID associated with the principal name. The following message contains the DCE status code text returned from the DCE API **sec_rgy_pgo_name_to_id**. In most cases, the reasons for the failure are:

- The principal name does not exist in the security registry.
- The Security daemon is not available.

The Display IMS Session Information request cannot be processed.

System Action: The program continues.

Administrator Response: Do one of the following, then try the request again:

- Specify a principal name that exists in the security registry.
- Use the DCE status code text in the following message to correct the error.

If the problem continues, contact the service representative.

ASUV00242A Application Support server cannot obtain IMS session information.

Severity: svc_c_sev_error

Explanation: The Application Support server cannot obtain IMS session information for the principal. The following message contains the reason for the failure. Additional diagnostic information may be logged in the console log. In most cases, the reasons for the failure are:

- The Application Support server is not an Application Support IMS server.
- There is no IMS Session Information for the principal.
- The Application Support server detects an error while obtaining the IMS Session Information.

System Action: The program continues.

Administrator Response: Do one of the following, then try the request again:

- Specify an Application Support IMS server.
- Use the following message to correct the error.

If the problem continues, contact the service representative.

ASUV00243I No IMS session information is found for principal.

Severity: svc_c_sev_error

Explanation: The principal does not have an open session for the Application Support server. No information is displayed.

System Action: The program continues.

ASUV00244A Application Support IMS server detects error that is not expected.

Severity: svc_c_sev_error

Explanation: Additional diagnostic information is logged in the console log.

System Action: The program continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00245A CICS EXCI call Initialize_User detects error loading module.

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility calls the external CICS interface (EXCI) Initialize_User. The call fails with response and reason codes that indicate that an error occurred while trying to load a CICS load module. The Application Support CICS Attach Facility is stopped.

System Action: The program continues with reduced function.

System Programmer Response: Correct the error using the response and reason codes and try the program again.

ASUV00252A User ID must be entered.

Severity: svc_c_sev_error

Explanation: The administrator pressed the Enter key from the Application Support administration program Perform DCE Login panel without specifying a principal name.

System Action: The program continues.

Administrator Response: Specify a principal name and try the request again.

ASUV00253A Password must be specified.

Severity: svc_c_sev_error

Explanation: The administrator pressed the Enter key from the Application Support administration program Perform DCE Login panel without specifying a password.

System Action: The program continues.

Administrator Response: Specify the password for the principal and try the request again.

ASUV00254A Specify server definition type: 1 for CICS, 2 for ISC, 3 for APPC, 4 for OTMA.

Severity: svc_c_sev_error

Explanation: When defining an Application Support server definition, the administrator must specify 1 for an Application Support CICS server, 2 for an Application Support IMS server ISC connection, 3 for an Application Support IMS server APPC connection, 4 for an Application Support IMS server OTMA connection.

System Action: The program continues.

Administrator Response: Specify an Application Support server type and try the request again, or enter PF3 to exit from the panel.

ASUV00255A Action code must be 0 or 1.

Severity: svc_c_sev_error

Explanation: The Application Support administration program Install/Uninstall Interfaces panel requires an action code for each interface. Valid actions are 0 to uninstall an interface, and 1 to install an interface.

System Action: The program continues.

Administrator Response: Select a valid action for each interface and try the request again.

ASUV00256A Interface name specified is not valid.

Severity: svc_c_sev_error

Explanation: The Install/Uninstall Interfaces panel of the Application Support administration program requires an interface name to be specified for each action entered in an Action field.

System Action: The program continues.

Administrator Response: Specify an interface name for every selected action and try the request again.

ASUV00257A Option specified is not valid. Choose 1 or 2.

Severity: svc_c_sev_error

Explanation: Options 1 or 2 is valid. Either an option outside of this range, or no option is specified.

System Action: The program continues.

Administrator Response: Select a valid option and try the request again, or press the PF3 key to exit from the panel.

ASUV00258A Option specified is not valid. Choose 1 to 3.

Severity: svc_c_sev_error

Explanation: Options 1 through 3 are valid. Either an option outside of this range, or no option is specified.

System Action: The program continues.

Administrator Response: Select a valid option and try the request again, or press the PF3 key to exit from the panel.

ASUV00259A You must enter an IMS principal name.

Severity: svc_c_sev_notice

Explanation: Enter which IMS principal name you wish to display.

System Action: The program continues.

**ASUV00400A Application Support IMS cannot release storage. function=*function*, tcb=*tcb*.
uuid=<cor-id>,line=<line_number>,file=<file>**

Severity: svc_c_sev_error

Explanation: The Application Support IMS function *function* cannot release 24 bit storage using the LE function CEEDSHP. This error should not occur during normal processing.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00401A Application Support IMS cannot destroy mutex, function, mutex, return_code, tcb. error_text
uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support IMS function *function* cannot destroy a mutex that is no longer needed. Additional information is given by return code *return_code* and the C library *error_text*, which are returned from the **pthread_mutex_destroy()** routine.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00402A Application Support IMS cannot create thread, function, thread, return_code, tcb. error_text
uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support IMS function *function* cannot create the thread *thread*. Application Support IMS Attach Facility is stopped. Additional information is given by return code *return_code* and the C library *error_text*, which are returned from the **pthread_create()** routine.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00403A Application Support IMS cannot destroy thread, function, thread, call, return_code, tcb. error_text
uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support IMS function *function* cannot destroy the thread *thread*. The Application Support IMS server attempts to destroy the thread by calling one of two C library functions:

- pthread_detech
- pthread_join

Additional information is given by return code *return_code* and the C library *error_text*, which are returned from the C library routine.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00404I Application Support IMS cannot open log file, function, file_name, tcb. error_text
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support IMS function *function* cannot open the Application Support IMS log (ASUVPLOG) file *file_name*. In most cases, the reasons for the failure are:

- The ASUVPLOG DD card is missing or incorrect in the startup procedure.
- The Application Support server does not have authority to open the file *file_name* for write access.

Unsolicited IMS messages are written to the log file and the messages are lost. Additional information about the error is given by the C library *error_text*, which is returned from the **fopen()** routine.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

System Programmer Response: Ensure that the Application Support server name is correct and proper authority is given to the Application Support server to open the file. Try the program again.

ASUV00405I Application Support IMS cannot close log file, function, file_name, tcb. error_text
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support IMS function *function* cannot close the Application Support IMS log (ASUVPLOG) file *file_name*. Additional information about the error is given by the C library *error_text*, which is returned from the **fclose()** routine.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

System Programmer Response: Try the request again. If the problem continues, contact the service representative.

ASUV00406A Application Support IMS cannot write to log file, function, file_name, tcb.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support IMS function *function* cannot write to the Application Support IMS log file *file_name*. In most cases, the reason for the failure is: that the Application Support server does not have the authority to write to the file.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

System Programmer Response: Grant the Application Support server write authority to the file and try the request again. If the problem continues, contact the service representative.

ASUV00407I Application Support IMS log file is full, wrapping to beginning of file, *function*, *file_name*, *tcb*.
userid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support IMS server detects that the log file *file_name* is full. The logwriter logs new IMS records at the beginning of the file, writing over old records. The log file is defined by the ASUVPLOG ddname in the startup procedure. The log file should be large enough to ensure that the records that are overwritten are no longer needed.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00409I Application Support IMS cannot write to log file *file_name*. Logging is not enabled. *function*, *tcb*.
userid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_notice

Explanation: The Application Support IMS logs IMS messages to the *file_name* log file. The Application Support IMS logwriter function *function*, cannot write an initial entry to the log file, so it suspends further logging. No IMS unsolicited messages are logged. In most cases, the reasons for the failure are:

- The ASUVPLOG ddname is not defined correctly in the startup procedure.
- The log file is not correctly allocated. The log file must be allocated as a sequential dataset before the Application Support IMS server is started.

System Action: The program continues.

System Programmer Response: Define or allocate the log correctly. Try the request again.

ASUV00410A Application Support IMS Attach Facility fails because it cannot create VTAM structure *struct*.
function, *return_code*, *tcb*.
userid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: During an Application Support IMS Attach Facility Start request, the Application Support IMS server detects an error when creating the VTAM structure named *struct*. The Attach Facility start request fails.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues with reduced function.

System Programmer Response: Try the request again. If the problem continues, contact the service representative.

ASUV00411A Application Support IMS detects error because it cannot call VTAM macro, *macro*,
parm1, *parm2*, *function*, *tcb*.
userid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support IMS Attach Facility detects an error when the function name *function* issues a VTAM macro call. The Attach Facility request fails. Refer to *OS/390 IBM Communications Server: SNA Programming*, SC31-8573, for an explanation of first parameter *parm1* and second parameter *parm2* for the following VTAM MACRO routines:

- MODCB, *parm1* = R15, *parm2* = R0.
- SHOWCB, *parm1* = R15, *parm2* = R0.
- SETLOGON, *parm1* = RPLRTNCD, *parm2* = RPLFDBK2.
- OPEN, *parm1* = R15, *parm2* is not used.
- CLOSE, *parm1* = R15, *parm2* is not used.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Operator Response: Ensure that IMS is running.

System Programmer Response: Try the request again. If the problem continues, contact the service representative.

ASUV00413A Application Support IMS detects interface routine error, *function*, *subpool*, *error_mod*, *error_id*, *return_code*, *tcb*.

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The function *function* call to the Application Support server for IMS fails. The *return_code* values are:

- 1 (Internal)**
Session entry block is not valid.
- 2 (Internal)**
Cannot obtain storage.
- 3 (Internal)**
Cannot free storage.
- 4 (Internal)**
Return code from a VTAM macro call is not valid.
- 5 (Internal)**
Completion of a VTAM macro operation is not valid.
- 6 (Internal)**
RPC request passed to IMS interface support is not valid.
- 7 (Internal)**
The SNA protocol sequence encountered during communication with IMS is not expected.
- 8** Configuration error. Application Support server must be defined as SINGLE1 VLVB node.
- 9** Stub is not valid. IMS response message does not match the size of the RPC parameters in the IDL definitions.
- 10 (Internal)**
Session contention is not expected. Asynchronous IMS traffic is received, but is not expected.
- 11** Session bind parameters received are not valid. The Application Support server must be defined as an ISC node supporting a negotiated BIND and parallel sessions.
- 12 (Internal)**
Entry state or ISC session state is not valid.
- 13 (Internal)**
IMS traffic is received but no valid user entry is found.
- 14 (Internal)**
Session fails.
- 15 (Internal)**
IMS closes the session.

The *error_mod* and *error_id* contain additional error symptom information. The *subpool* is the ISC session on which the error occurred.

Refer to messages ASUV00415A and ASUV00416A, if logged, for additional information and diagnostic data.

System Action: In all situations, the connection to IMS ends and the caller is notified. All subsequent calls from the same caller are rejected until a new IMS connection is established by a call that opens the ISC session.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Programmer Response: Correct the error based upon the return code *return_code*. Then try the request again. If the problem continues, contact the service representative.

Programmer Response: For the conditions that are not internal processing errors, fix the error, then try the request again.

ASUV00415A Application Support IMS detects VTAM error, function, subpool, vtam_req_id, vtam_ret_code, vtam_feedback_code, error_code.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: Application Support IMS detects an error returned from VTAM. Refer to *OS/390 IBM Communications Server: SNA Programming, SC31-8573*, for an explanation of the returned values. Also refer to messages ASUV00413A and ASUV00416A, if logged, for additional information and diagnostic data. The variables are:

- *vtam_req_id* is the VTAM operation code (RPL REQ field).
- *vtam_ret_code* is the VTAM return code (RPL RTNCD field).
- *vtam_feedback_code* is the VTAM feedback code (RPL FDB2 field).
- If *vtam_ret_code* is not 0, *error_code* is the SNA sense code.
- If *vtam_ret_code* = 0, *error_code* is the return code from the VTAM macro instruction.
- *subpool* is the ISC session on which the error occurred.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: In all situations, the connection to IMS ends and the caller is notified by a return code. All subsequent calls from the same caller are rejected until a new IMS connection is established by a call that opens the ISC session.

System Programmer Response: Correct the error based upon the returned values. Then try the request again. If the problem continues, contact the service representative.

Programmer Response: Establish a new IMS connection. Then try the request again.

ASUV00416A Application Support IMS creates internal trace, function, subpool, sequence, trace_data, tcb.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: One of the IMS interface routines finds an error that is not expected. Refer to messages ASUV00413A and ASUV00415A, if logged, for additional information and diagnostic data to report to the support representative. The internal trace also provides diagnostic information for the service representative. The variables are:

- *subpool* is the ISC session on which the error occurred.
- *sequence* is a message sequence number. This message is repeated to log all the trace information. The sequence number starts at 1 and is incremented each time this message is logged.
- *trace_data* is an internal function trace.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

System Programmer Response: If the problem continues, contact the service representative.

Programmer Response: Try the request again.

ASUV00420A Application Support IMS detects that adaptor state table is not consistent, *function*, *tcb*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support IMS function *function* determines that the state table is not in a consistent state.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00421A Application Support IMS Attach Facility stop request fails, *function*, *tcb*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* determines that stop attachment processing fails. Other related messages appear in the console log.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00422A Application Support IMS initiates Attach Facility stop, *function*, *tcb*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* finds an error from which it cannot recover, and it starts the Attach Facility stop processing.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues with reduced function.

Operator Response: Start the Application Support server again.

System Programmer Response: If the problem continues, contact the service representative.

ASUV00430I Application Support IMS opens ISC session. *function*, *node*, *subpool*, *dceid*, *mvsid*, *tcb*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_notice_verbos

Explanation: The Application Support IMS function *function* allocates an ISC session. The session is opened on node *node*, with subpool *subpool*. The session uses the DCE identity *dceid*.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00431I Application Support IMS closes ISC session. *function, node, subpool, tcb.*

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_notice_verbose

Explanation: The Application Support IMS function *function* closes an ISC session on behalf of DCE client. The session is closed on node *node*, with subpool *subpool*.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00432I Application Support IMS cancels call to IMS, *function, node, subpool, opdata, tcb.*

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_notice

Explanation: The Application Support server function *function* is processing a remote call to an IMS transaction and the processing is cancelled. The Application Support server VTAM LU Name is *node*. The ISC session name is *subpool*. The IMS operation data (transaction code) is *opdata*.

One thread in a client program stops the processing on another thread. The processing is stopped by:

- Cancelling the thread that is making the remote call.
- Closing the ISC session used by the other thread.
- Ending the client program.

System Action: The remote call to the IMS transaction is cancelled.

Programmer Response: If the thread is cancelled because the processing is suspended, refer to message ASUV00430I for more information. Then ask the IMS administrator to find the reason that the transaction processing does not complete.

ASUV00437I Null resource manager name returned from IMS.

Severity: svc_c_sev_error

Explanation: An IMS/ESA Version 6.1 system or higher using OTMA should return a recoverable resource manager name.

System Action: The Application Support server continues.

Operator Response: Update the file.

System Programmer Response: Update envar file.

ASUV00438E Variable *varname* missing from envar file, aborting startup.

Severity: svc_c_sev_error

Explanation: The named variable is missing, misspelled, or null in the envar file. A valid value for this variable is required to run the Application Support server.

System Action: The Application Support server ends.

Operator Response: Update the envar file.

System Programmer Response: Update envar file.

ASUV00439E The Application Support server module is not APF-authorized. Startup can not continue.**Severity:** svc_c_sev_error**Explanation:** The Application Support server module is running from a non-APF-authorized library. Recoverable Resource Management Services requires this.**System Action:** The Application Support server ends.**Operator Response:** Have the system programmer move the server module to an APF-authorized library.**System Programmer Response:** Move the server module to an APF- authorized library. Make any necessary JCL changes.

ASUV00440I *product_identifier***Severity:** svc_c_sev_notice**Explanation:** This is an informational message issued when the IMS server starts. It displays product name, release number, version, and the date and time the module was built by IBM: such as, '5647-A01 OS/390 DCE Application Support IMS OTMA Server 1997070913.2.' However, a server built with the Customization Component may contain different information, which was supplied by the VERSION function of the Customization Component.**System Action:** The program continues.

ASUV00441E Server environment variable _ASU_APF=NO was not coded and the Application Support server module is not APF-authorized. Startup can not continue.**Severity:** svc_c_sev_error**Explanation:** The Application Support server module is running from a non-APF-authorized library. The environment variable indicates an intent to run authorized.**System Action:** The Application Support server ends.**Operator Response:** Have the system programmer move the server module to an APF-authorized library.**System Programmer Response:** Move the server module to an APF- authorized library. Make any necessary JCL changes. The Application Support server can run unauthorized, by specifying _ASU_APF=NO as an environment variable, but client MVS userid can not be passed to IMS.

ASUV00442E RACROUTE test rc =racroute_rc reasons=racroute_reason RACF_rc RACF_reason RACROUTE_rc RACROUTE_reason**Severity:** svc_c_sev_error**Explanation:** The Application Support server module received an error from the first access to RACF or other security subsystem. Reasons are: return code from RACROUTE, RACF return code, RACF reason code, RACROUTE return code, and RACROUTE reason code. If RACF fails at this point, the most likely cause is missing or improper RACF profiles.**System Action:** The Application Support server ends.**Operator Response:** Have security profiles checked**System Programmer Response:** Check RACF return codes

ASUV00443E STARTATT command failed reason=reason_code**Severity:** svc_c_sev_error**Explanation:** The start attachment command (STARTATT) failed. The STARTATT command was issued by the ASUADMIN program or the system operator or by specifying _ASU_AUTOSTART=YES in the Application Support server envvar file. For an explanation of the reason codes refer to the List of Common Return Code Values in *z/OS DCE Application Support Programming Guide*, SC24-5902. For example, for the APPC or OTMA adaptors with _ASU_EXIT=AUTO in the Application Support server

envar file, reason 90 = Authority insufficient to communicate with IMS and 91 = APPC communication error. For other adaptors, look for other messages to explain the error.

System Action: The Application Support server continues.

Operator Response: Correct the problem and issue STARTATT command using the MVS MODIFY system command or the ASUADMIN program.

System Programmer Response: If the cause is not found, run with SVC_VEN_DBG=*.9 to obtain diagnostic messages.

ASUV00444E _ASU_SYMBOLIC_NAME missing from envar file, aborting startup.

Severity: svc_c_sev_error

Explanation: _ASU_SYMBOLIC_NAME=appc_symbolic_name_of_IMS must be specified in the envar file used by the Application Support server.

System Action: The Application Support server ends.

Operator Response: Have envar file updated.

System Programmer Response: Update envar file

ASUV00445E _ASU_SERVER_LUNAME missing from envar file, aborting startup.

Severity: svc_c_sev_error

Explanation: _ASU_SERVER_LUNAME=vtam_lu_name to be used by the server in APPC conversations with IMS. This parm must be specified in the envar file used by the Application Support server.

System Action: The Application Support server ends.

Operator Response: Have envar file updated.

System Programmer Response: Update envar file

ASUV00446I varname is invalid. The default default is used.

Severity: svc_c_sev_error

Explanation: An invalid number has been coded for the environmental variable in the envar file used by the Application Support server.

System Action: The Application Support server continues.

Operator Response: Have envar file updated.

System Programmer Response: Update envar file

ASUV00447I variable is invalid. The default default is used.

Severity: svc_c_sev_error

Explanation: An invalid value has been coded for the environmental variable in the envar file used by the Application Support server.

System Action: The Application Support server continues.

Operator Response: Have envar file updated.

System Programmer Response: Update envar file

ASUV00449I No MVS user ID found for principal *dce_name*.

Severity: svc_c_sev_error

Explanation: There was no entry in the mapping file or RACF for the client DCE principal.

System Action: The client session is refused.

Operator Response: none.

System Programmer Response: Update mapping file if DCE principal is to have access to Application Support server sessions or investigate user's unauthorized attempts to access Application Support server.

ASUV00489I Application Support OTMA cannot attach to IMS.

Severity: svc_c_sev_error

Explanation: In most cases, the reasons for the failure are:

- The XCF-defined connection to the IMS system from the server is improperly configured.
- IMS is not running.
- IMS is running but with OTMA stopped.

Additional diagnostic information is available from the dump and probe of user abend xBBF. Refer to message ASUV00933A for CICS information associated with this failure.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Ensure that the IMS system associated with the Application Support server is operational and has OTMA started. Then try the request again. If the problem continues, save the dump and contact the service representative.

ASUV00500A Application Support server cannot allocate storage, *function*, *tcb*.

userid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot allocate storage using the **malloc()** function. This failure may result from a region size that is too small or from storage that is not released. This error should not occur during normal processing.

For more information about the variables in this message, see "Variables Common To All Messages" on page 195.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Ensure that sufficient memory is available, then try the request again. Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00501A Application Support server cannot initialize mutex, *function*, *mutex*, *return_code*, *tcb*, *error_text*.

userid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot initialize the mutex *mutex*. Additional information about the error is given by the C library *error_text*, which is returned from the **pthread_mutex_init()** routine. This error should not occur during normal processing.

For more information about the variables in this message, see "Variables Common To All Messages" on page 195.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00502I Application Support server detects error that is not expected, function, exception, sys, reason, service_data, tcb.

uid=<cor-id>,line=<line_number>,file=<file>.

Severity: svc_c_sev_notice

Explanation: The Application Support server function *function* detects an exception *exception* that is not expected. If the exception contains EXC, then *sys* contains the MVS system completion code, and *reason* contains the MVS reason code. Otherwise, *exception* contains the failing function call, *sys* contains the function call return code, and *reason* is not used.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

Programmer Response: Determine if the exception is caused by user action, for example the cancellation of a unit of work, or another problem. Then try the request again.

ASUV00503A Application Support IMS cannot lock mutex, function, mutex, return_code, tcb. error_text

uid=<cor-id>,line=<line_number>,file=<file>.

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot lock the mutex *mutex*. Mutexes are used to protect system resources. Additional information about the error is given by the C library *error_text*, which is returned from the **pthread_mutex_lock()** routine.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00504A Application Support server cannot unlock mutex, function, mutex, return_code, tcb. error_text

uid=<cor-id>,line=<line_number>,file=<file>.

Severity: svc_c_sev_error

Explanation: The function *function* cannot unlock the mutex *mutex*. Mutexes are used to protect system resources. Additional information about the error is given by the C library *error_text*, which is returned from the **pthread_mutex_unlock()** routine.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: If the problem continues, save the dump and contact the service representative. Use the dump to help determine the probable cause of the failure.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00506A Application Support IMS ACB password is too long, function, parameter, tcb.

userid=<cor-id>,line=<line_number>,file=<file>.

Severity: svc_c_sev_fatal

Explanation: The Application Support server function *function* detects that the Access Method Control Block (ACB) password parameter *parameter* is not correct. The ACB password is an optional input parameter to the Application Support IMS server. The ACB password must be between 1 and 8 characters long.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: If the input parameter problem is caused by a user error, correct the error and try again.

ASUV00510A Application Support IMS detects that data structure structure is not correct, function, tcb

userid=<cor-id>,line=<line_number>,file=<file>.

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* determines that the data structure *structure* is not correct or is damaged.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00511A Application Support IMS cannot address control structure, function, structure, tcb.

userid=<cor-id>,line=<line_number>,file=<file>.

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot address a critical control structure *structure*.

For more information about the variables in this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00512A Application Support IMS detects internal error trying to serialize VTAM open, close: *function, message_severity, message_number, return_code, tcb.*

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: Application Support IMS detects an internal error when trying to serialize VTAM Open or Close.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00514I Application Support server cannot open memory log, log_name. DCE status code: error_status.

Severity: svc_c_sev_notice

Explanation: The Application Support server cannot open memory log *log_name*.

System Action: The Application Support server should recover and continue operating but there will be no memory log.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. If the problem continues, contact the service representative.

ASUV00515A Application Support IMS cannot clean up IMS interface resources. Return code=return_code.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: At the end of the IMS session, the Application Support server tried to reset global variables.

System Action: Future transactions may not be able to be performed.

System Programmer Response: Try the request again. If the problem continues, contact the service representative.

ASUV00550I Application Support server is initializing.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_notice_verbose

Explanation: The Application Support server is starting the initialization process.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00551A Application Support server cannot process the input parameters because they are invalid. *function.*

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot process the server input parameters. Application Support server/CICS expects two input parameters, Application Support server entry name and Application Support server principal name. Application Support server/IMS expects three input parameters, Application Support server entry name, Application Support server principal name, and VTAM password.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Start the Application Support server again with the correct input parameters.

ASUV00553A Application Support server principal ID is not valid, *function*,ID=<principal_ID>.
uid=<cor-id>,line=<line_number>,file=<file>.

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot log in to DCE with the principal identifier *principal_ID*. The principal ID, which is an input parameter specified in the JCL to start the Application Support server, is not valid.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Start the Application Support server again with correct Application Support server principal name.

ASUV00554A DCE Security service cannot set up network identity for Application Support server, *function*.

DCE status code: *status_code*.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* issues the **sec_login_setup_identity** call as part of the DCE login procedure to set up the network identity of the Application Support server. The call to **sec_login_setup_identify** fails, and the DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: Processing of the current request ends.

Administrator Response: Ensure that the DCE security service is operational. Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the program again.

ASUV00555A DCE Security service cannot obtain Application Support server password, *function*. DCE status code: *status_code*.

uid=<cor-id>,line=<line_number>,file=<file>.

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* issues the **sec_key_mgmt_get_key** call to the DCE security component as part of the DCE login procedure to obtain the server password. The call to **sec_key_mgmt_get_key** fails and the DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The current request ends.

Administrator Response: Check that the Application Support server principal is set up correctly in the DCE security database. Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the program again.

ASUV00556A Application Support server password is expired, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* cannot log in to DCE with the principal ID because the Application Support server password is expired.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The current request ends.**Administrator Response:** Reset the Application Support server password. Then try the request again.

ASUV00557A Cannot create network credential, *function*. DCE status code: *status_code*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The function *function* issues the **sec_login_set_context** call to the DCE security component as part of the DCE login procedure to create a network credential. The call to **sec_login_set_context** fails and the return code is *status_code*. The likely cause of the problem is that the DCE security component not operational.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The current request ends.**Operator Response:** Ensure that the DCE security component is operational.**System Programmer Response:** Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Then try the request again.

ASUV00558A Application Support server cannot validate login context with network authority, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* cannot validate and certify a login context as part of the DCE login procedure because the login context is only locally authenticated. The most likely cause of the problem is that the DCE security component is not operational.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends. The login context cannot be used to establish network credentials.**Operator Response:** Ensure that the DCE Security component is operational. Then try the request again.

ASUV00559A Application Support server cannot validate or certify login context, *function*. DCE status code: *status_code*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* issues the **sec_login_valid_and_cert_ident** call to the DCE security component as part of the DCE login procedure. The call to **sec_login_valid_and_cert_ident** fails and the DCE status code is *status_code*. The most likely cause of the problem is that the DCE security component is not operational.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The current request ends.

Operator Response: Ensure that the DCE security component is operational.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Then try the request again.

ASUV00561A Application Support server detects exception that is not expected, *function*, *sys_code*, *reason_code*, *tcb*.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* catches an exception that is not expected. This error should not occur during normal processing. see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues. Resources allocated for this request are freed so that the server can process other requests.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Check system code *sys_code* and reason code *reason_code*. Ensure that the exception is not caused by a cancellation from the client. Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00562A Application Support server cannot update Application Support server password, *function*. DCE status code: *status_code*.

uid=<cor-id>,line=<line_number>,file=<file>.

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* issues the **sec_key_mgmt_manage_key** call to the DCE Security component to update the Application Support server password. This is part of the DCE login procedure during Application Support server initialization. The call to **sec_key_mgmt_manage_key** fails and the DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again.

ASUV00563I Application Support server is logging in to DCE.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_notice_verbose

Explanation: The Application Support server is logging in to DCE during Application Support server initialization.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00564A Application Support server is not authorized to obtain password, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* issues the **sec_key_mgmt_get_key** call to the DCE Security component to obtain the password of the server. This is part of the DCE login procedure during Application Support server initialization. The call to **sec_key_mgmt_get_key** fails because the Application Support server program is not authorized to obtain the password from the DCE Security component.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The current request ends.**Administrator Response:** Authorize the Application Support server program to obtain the password. Then try the request again.

ASUV00565A Application Support server cannot create protocol sequence, *function*. DCE status code:*status_code*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* issues the **rpc_server_use_all_protseq** call to the RPC run time to create a protocol sequence. This is part of the Application Support server initialization. The call to **rpc_server_use_all_protseq** fails and the DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**System Programmer Response:** Ensure that the communication support, such as TCP/IP, is operational. Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Then try the request again.

ASUV00566I Application Support server binding, *binding*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_notice_verbose**Explanation:** The Application Support server logs the binding *binding* for each protocol sequence created during server initialization.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00567I Application Support server is registering management interface with local endpoint map and RPC runtime.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_notice_verbose**Explanation:** The Application Support server logs this message during Application Support server initialization.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00568A Application Support server entry name has invalid syntax, *function*, *entry_name*.

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot obtain configuration parameters from the directory because the name *entry_name* of the directory entry contains a syntax that is not valid.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Start the Application Support server again with the correct principal name for the Application Support server.

ASUV00569A Application Support server call to expand Application Support server entry name fails, *function*.

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* call to expand the Application Support server entry name fails.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Operator Response: Ensure that the DCE Directory service is operational.

System Programmer Response: Try the request again. If the problem continues, contact the service representative.

ASUV00570A Application Support server cannot obtain Application Support server bindings, *function*. DCE

status code: *status_code*.

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* call to the **rpc_server_inq_bindings** API during initialization cannot obtain all the server bindings created by the RPC run time. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00571A Application Support server cannot obtain Application Support server string bindings, *function*.

DCE status code: *status_code*.

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* call to the **rpc_server_binding_to_string_binding** API cannot obtain the Application Support server string binding that is created by the RPC run time during initialization. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00572A Application Support server cannot free RPC string, function. DCE status code: *status_code*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* call to the **rpc_string_free** API cannot free an RPC string created by the RPC run time. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00573A Application Support server cannot register interface with local endpoint map function. DCE status code: *status_code*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot register an interface with the local endpoint map by calling the **rpc_ep_register** API. The DCE status code is *status_code*. The DCE Host daemon is not operational or a temporary network problem exists.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Operator Response: Ensure that the DCE Host daemon is operational.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00574A Application Support server cannot register interface with RPC runtime, function. DCE status code: *status_code*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot register an interface with the RPC run time by calling the **rpc_server_if_register** API. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00575A Application Support server stub table is damaged, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* detects that the Application Support server stub table is damaged.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and ends abnormally.**Operator Response:** Collect the dump and probe information and notify the system programmer.**System Programmer Response:** Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00576A Application Support server cannot export interface to directory, *function*. DCE status code: *status_code*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* cannot export an interface to a directory by calling the **rpc_ns_binding_export** API. The DCE status code is *status_code*. Either the DCE Directory service is not operational or there is a temporary network problem.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**Operator Response:** Ensure that the DCE Directory service is operational.**System Programmer Response:** Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00577A Application Support server is ending because initialization fails, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* stops the server because of an error during server initialization.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**System Programmer Response:** Check the console log for messages that occur prior to this message to determine the reason for failure. Start the Application Support server again. If the problem continues, contact the service representative.

ASUV00578A Application Support server is ending because of error, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* is stopping the server because of an error. The probe and dump provide additional information about the failure.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and ends abnormally.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00579I Application Support server starts listening, *number_of_threads*.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_notice

Explanation: The Application Support server is ready to accept remote procedure calls. This message is displayed either when the server initialization is complete or when the Attach Facility starts or stops. The number of concurrent calls that the server can process is *number_of_threads*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00580A Application Support server cannot accept RPC requests, *function*. DCE status code: *status_code*.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The RPC run time stops accepting remote procedure calls. The error is detected by Application Support server function *function*. When the RPC run time stops listening, the **rpc_server_listen** routine returns with a DCE status code of *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00587A Application Support server cannot stop RPC run time listener, *function*. DCE status code: *status_code*.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* call to the **rpc_mgmt_stop_server** API cannot stop the RPC run time listener. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00588A Application Support server cannot obtain client authentication information, *function*. DCE status code: *status_code*.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* call to the **rpc_binding_inq_auth_client** API cannot obtain a privilege attribute certificate of a client. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00589A Application Support server internal error occurs, *function*, *tcb*.

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* detects an internal error in the Application Support server. This error should not occur during normal processing.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00593A Application Support server cannot register authentication information, *function*. DCE status

code: *status_code*.

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* call to the **rpc_register_auth_info** API cannot register the authenticated information with the RPC run time. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00595A Application Support server cannot unregister interface from the local endpoint map, *function*, *tcb*.

DCE status code: *status_code*.

uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* call to the **rpc_ep_unregister** API cannot unregister an interface. This problem can occur when the DCE Host daemon or the network is not operational. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00596A Application Support server cannot register transaction interface with RPC run time, *function*, *tcb*.**DCE status code:** *status_code*.**uid**=<*cor-id*>, **line**=<*line_number*>, **file**=<*file*>**Severity:** *svc_c_sev_error***Explanation:** The Application Support server function *function* cannot register the transaction interface with the RPC run time. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**System Programmer Response:** Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00597A Application Support server cannot unregister transaction interface from RPC run time, *function*, *tcb*, *state*. DCE status code: *status_code*.**uid**=<*cor-id*>, **line**=<*line_number*>, **file**=<*file*>**Severity:** *svc_c_sev_error***Explanation:** The Application Support server function *function* cannot unregister transaction interface from the RPC run time. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**System Programmer Response:** Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00598A Application Support server cannot unregister interface from RPC endpoint mapper, *function*.**DCE status code:** *status_code*.**uid**=<*cor-id*>, **line**=<*line_number*>, **file**=<*file*>**Severity:** *svc_c_sev_error***Explanation:** The Application Support server function *function* cannot unregister an interface from the RPC run time. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**System Programmer Response:** Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00600A Application Support server return code from Attach Facility is not defined, *function*, *status_code*.**uid**=<*cor-id*>, **line**=<*line_number*>, **file**=<*file*>**Severity:** *svc_c_sev_error***Explanation:** The Application Support server function *function* receives a status code of *status_code* from the Attach Facility during a transaction invocation. The return code is not expected.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues.**Operator Response:** Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00601A Application Support server is not authorized to update password, function.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* issues the **sec_key_mgmt_manage_key** call to the DCE security component to change the password of the server before it expires. The Application Support server is not authorized to perform this operation.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Administrator Response: Authorize the Application Support server to update the password. Start the Application Support server and try the request again. If the problem continues, contact the service representative.

ASUV00602A Application Support server cannot test if CDS attribute value exists, function, lstatus, rstatus.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot test if a CDS attribute value exists.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Operator Response: Ensure that the DCE Directory service is operational.

System Programmer Response: Try the program again. If the problem continues, contact the service representative.

ASUV00603A Application Support server cannot add a value to a CDS attribute, function, lstatus, rstatus.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: Application Support server cannot add a value to a CDS attribute.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Operator Response: Ensure that the DCE Directory service is operational.

System Programmer Response: Try the program again. If the problem continues, contact the service representative.

ASUV00604A Application Support server cannot remove CDS attribute, function, lstatus, rstatus.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: Application Support server cannot remove the CDS attribute value from the entry.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Operator Response: Ensure that the DCE Directory service is operational.

System Programmer Response: Try the program again. If the problem continues, contact the service representative.

ASUV00605A Application Support server cannot read the CDS attribute, *function*, *lstatus*, *rstatus*.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: Application Support server cannot read the CDS attribute from the DCE Directory service.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Operator Response: Ensure that the DCE Directory service is operational.

System Programmer Response: Try the program again. If the problem continues, contact the service representative.

ASUV00606A Application Support server cannot obtain a handle to CDS entry, *function*.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: Application Support server cannot obtain a handle to a CDS entry.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Operator Response: Ensure that the DCE Directory service is operational.

System Programmer Response: Try the program again. If the problem continues, contact the service representative.

ASUV00607A Application Support server detects attribute length that is not expected, *function*, *attr_name*, *exp_len*, *read_len*.

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot read the attribute *attr_name* from the directory because the length of attribute name *attr_name* is not expected. The expected length is *exp_len*, and *read_len* is the attribute read length. This error should not occur during normal processing.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Operator Response: Ensure that the DCE Directory service is operational.

System Programmer Response: Ensure that the CDS directory is not damaged. Try the program again. If the problem continues, contact the service representative.

ASUV00608A Application Support server cannot free server key, function. DCE status code: *status_code*.
uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: Application Support server function *function* cannot free a server key that is either NULL or is not valid. This error should not occur during normal processing. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00609A Application Support server main control block is damaged, function.
uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* detects a damaged server main control block. This error should not occur during normal processing.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00611A Application Support server cannot unexport interface from directory, function. DCE status code: *status_code*.
uid=<cor-id>,line=<line_number>,file=<file>

uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: Application Support server function *function* cannot unexport an interface from the DCE directory. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00613I Application Support server is reading configuration from directory.
uid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_notice_verbose

Explanation: Application Support server is reading a configuration from the DCE directory during initialization.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00614A Application Support server function *function* cannot read Application Support server configuration from directory.

uuid=<cor-id>, **line**=<line_number>, **file**=<file>

Severity: svc_c_sev_error

Explanation: Application Support server function *function* cannot read the server configuration from the directory. In most cases, the reasons for the failure are:

- The DCE directory is not operational.
- An Application Support server entry is not created correctly.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Administrator Response: Ensure that the DCE Directory service is operational and the Application Support server entry is created correctly. Try the request again. If the problem continues, contact the service representative.

ASUV00615A Application Support server function *function* cannot read Application Support server object UUID. DCE status code: *status_code*.

uuid=<cor-id>, **line**=<line_number>, **file**=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot read the Application Support server object UUID from the directory. The DCE status code is *status_code*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00616A Application Support server cannot read CDS attribute because it does not exist, *function*.

uuid=<cor-id>, **line**=<line_number>, **file**=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* tries to read a CDS attribute, but the attribute does not exist.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Administrator Response: Ensure that the CDS attribute exists and try the request again. If the problem continues, contact the service representative.

ASUV00617A Application Support server password is not in server keytab file, *function*.

uuid=<cor-id>, **line**=<line_number>, **file**=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot obtain the server password from the server keytab file. In most cases, the reason for the failure is: the server password is not set in the server keytab file.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Administrator Response: Ensure that the password is set in the server keytab file. Try the request again. If the problem continues, contact the service representative.

ASUV00618A Application Support server entry does not exist in the CDS directory, *function*, *server_entry*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The function *function* tries to query the server entry *server_entry*, but it is not in the CDS directory. The Application Support server ends because it cannot obtain the required configuration parameters from the CDS directory. In most cases, the reason for the failure is: that the server entry is not created in the CDS directory.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Administrator Response: Ensure that the server entry is in the CDS directory. Then try the request again. If the problem continues, contact the service representative.

ASUV00619I Application Support server is exporting management interface to directory.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_notice_verbose

Explanation: Application Support server is exporting the management interface to the directory during initialization.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program continues.

ASUV00620A Application Support server is not authorized to update directory, *function*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: Application Support server function *function* tries to update the server entry in the directory, but the server program is not authorized to perform the update operation. In most cases, the reason for the failure is: that DCE access control is not correctly set for the server principal ID.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Administrator Response: Ensure that the DCE access control is correctly set for the server principal ID. Then try the request again. If the problem continues, contact the service representative.

ASUV00622A Application Support server object UUID is not valid, *function*.
uuid=<cor-id>,line=<line_number>,file=<file>

Severity: svc_c_sev_error

Explanation: The Application Support server function *function* cannot read the server object UUID from the DCE directory because the format of the object UUID is not valid. The error occurs during server initialization. In most cases, the reason for the failure is: a damaged or incorrectly created server entry.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.

Administrator Response: Ensure that the server entry is not damaged and is created correctly. Then try the request again. If the problem continues, contact the service representative.

ASUV00623A Application Support server cannot install server stub, *function*, *stub*.**uid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* cannot install the server stub *stub* during server initialization.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**Administrator Response:** Ensure that the stub exists and try the request again. Ensure the stub library is APF-authorized if the AS IMS server module runs from an APF-authorized library. If the problem continues, contact the service representative.

ASUV00624A Application Support server cannot read installed interface from directory, *function*.**uid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* cannot read the installed interfaces from the CDS directory during server initialization. In most cases, the reason for the failure is: that the server entry is not created correctly.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**Administrator Response:** Ensure that the server entry is created correctly. Then try the request again. If the problem continues, contact the service representative.

ASUV00625A Application Support server detects CDS class version that is not valid, *function*, *major*, *minor*.**uid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* cannot read the server entry class version from the directory because the class version is not valid. The major version is *major* and the minor version is *minor*. The expected major version is 1.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**Administrator Response:** Ensure that the server entry class version is correct. Then try the request again. If the problem continues, contact the service representative.

ASUV00626A Application Support server detects CDS class that is not valid, *function*, *class*, *exp_class*.**uid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* detects a CDS class *class* from the directory that is not valid. The expected class is *exp_class*.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**Administrator Response:** Ensure that the server entry is correct. Then try the request again. If the problem continues, contact the service representative.

ASUV00627A Application Support server detects error in standard input data, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* reads the input from standard input. In most cases, the reason for the failure is: a C/C++ run time error.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**System Programmer Response:** Ensure that there is no C/C++ run time error and try the request again. If the problem continues, contact the service representative.

ASUV00628A Application Support server cannot initialize Attach Facility, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* cannot initialize the Attach Facility.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**Administrator Response:** Review previous messages and correct any errors.**System Programmer Response:** Check the console log for previous messages to determine the reason for failure, and correct the error based on the message text. Then try the request again. If the problem continues, contact the service representative.

ASUV00630A Application Support server subsystem ID is not valid, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** The Application Support server function *function* cannot initialize the Attach Facility because the subsystem data is not valid. In most cases, the reason for the failure is: that the subsystem data in the directory is not correct.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program ends.**Administrator Response:** Ensure that the subsystem data in the directory is correctly set. Then try the request again. If the problem continues, contact the service representative.

ASUV00631A Application Support server Attach Facility cannot obtain storage, *function*.**uuid=<cor-id>,line=<line_number>,file=<file>****Severity:** svc_c_sev_error**Explanation:** Application Support server function *function* is initializing the Attach Facility and the Attach Facility cannot obtain storage. This error should not occur during normal processing.

For more information about the variables of this message, see “Variables Common To All Messages” on page 195.

System Action: The program writes a dump and ends abnormally.**Operator Response:** Collect the dump and probe information and notify the system programmer.**System Programmer Response:** Increase the region size for the Application Support server and try the request again. Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00632A Application Support server cannot configure Access Control List manager.

DCE status code: *status_code*.

Severity: *svc_c_sev_error*

Explanation: The Application Support server cannot configure the Access Control List manager. The DCE status code is *status_code*.

System Action: The program ends.

Administrator Response: Ensure that all **acl.dat* files exist and are valid. For more information on the **acl.dat* files, refer to *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903. Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00633A Application Support server cannot define PRINTSTRINGS for access control.

DCE status code: *status_code*.

Severity: *svc_c_sev_error*

Explanation: An internal error occurs when trying to define PRINTSTRINGS for the Application Support server. The DCE status code is *status_code*.

System Action: The Access Control List database file may not be generated and the Application Support server may terminate.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00634A Application Support server cannot export interface.

DCE status code: *status_code*.

Severity: *svc_c_sev_error*

Explanation: This error can occur because the interface is not found or the Application Support server was not authorized to export the interface to CDS or the endpoint map.

System Action: The program continues.

System Programmer Response: Verify that the DCE principal for your Application Support server has permission to read and write to the CDS object (see *Configuring the CDS Namespace Entries* in the *z/OS DCE Application Support Configuration and Administration Guide*). Also verify that the Application Support server can access the DCE endpoint map (see *Creating the DCE Account for the Application Support server* in the same book).

ASUV00635A User is not authorized to perform this operation.

Severity: *svc_c_sev_error*

Explanation: The user does not have the authority to perform the operation.

System Action: The program continues with reduced function.

Administrator Response: Ensure that the user has sufficient authority for the request. Try the request again.

ASUV00636A Application Support server cannot obtain expiration time for login context. DCE status code: *status_code*.

Severity: svc_c_sev_error

Explanation: The Application Support server cannot obtain the expiration time for the login context. The DCE status code is *status_code*.

System Action: The program ends.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00637A Application Support server cannot refresh login context.

DCE status code: *status_code*.

Severity: svc_c_sev_error

Explanation: The Application Support server cannot refresh the login context. The DCE status code is *status_code*.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00638A Application Support server cannot renew login context.

DCE status code: *status_code*.

Severity: svc_c_sev_error

Explanation: The Application Support server cannot renew the login context. The DCE status code is *status_code*.

System Action: The program ends.

System Programmer Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00700A Application Support server cannot open identity mapping file.

Severity: svc_c_sev_error

Explanation: An error occurs when attempting to open the identity mapping file.

System Action: The program ends.

Administrator Response: Verify that the `_ASU_IDMAP` environment variable is defined, and points to an identity mapping file. Then try the operation again.

ASUV00701A Application Support server identity mapping file is damaged.

Severity: svc_c_sev_error

Explanation: The Application Support server identity mapping file is damaged.

System Action: The program ends.

System Programmer Response: Ensure that the `_ASU_IDMAP` environment variable contains the name of the current identity mapping file. If the problem continues, contact the service representative.

ASUV00702I Duplicate mapping for principal '*principal*' and server name '*server*' found in the identity mapping file.

Severity: svc_c_sev_warning

Explanation: Another entry defining a mapping for the specified principal and server name already exists.

System Action: The duplicate entry is skipped and processing of the identity mapping file continues.

Administrator Response: Delete the duplicate entry from the input file and try the operation again.

ASUV00703I Invalid MVS identity '*identity*' found during identity mapping file processing.

Severity: svc_c_sev_warning

Explanation: An invalid MVS ID, *identity*, was found in the identity mapping file.

System Action: Processing of the identity mapping file continues.

Administrator Response: Ensure that the MVS ID is not longer than 8 characters and try the operation again.

ASUV00704I Cannot translate DCE identity '*dce_identity*' to principal and cell UUIDs during identity mapping file processing.

Severity: svc_c_sev_warning

Explanation: The DCE identity could not be translated into the principal and cell UUIDs.

System Action: Processing of the identity mapping file continues.

Administrator Response: Verify that the DCE identity exists, and has the correct format. Try the operation again. If the problem continues, contact the service representative.

ASUV00705A Application Support identity mapping table cannot be created from identity mapping file.

Severity: svc_c_sev_error

Explanation: The identity mapping table used by the Application Support server cannot be created because of previously reported errors found during identity mapping file processing.

System Action: The program ends.

Administrator Response: Correct the errors reported previously in the identity mapping input file. For more information on the Application Support identity mapping function, refer to *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903. Try the operation again.

ASUV00706A Unexpected error detected during Application Support identity mapping file processing.

Severity: svc_c_sev_error

Explanation: An error occurred in the processing of the identity mapping file that was not expected.

System Action: The program ends.

Administrator Response: Ensure that the identity mapping file is correct and try the operation again. If the problem continues, contact the service representative.

ASUV00707A Cannot open Application Support server identity mapping file '*file_name*'. *error_text*.

Severity: svc_c_sev_error

Explanation: The identity mapping file *file_name* cannot be opened. Ensure that the file exists and that it is correctly specified. Additional information about the error is given by the C library *error_text*, which is returned from the **fopen()** routine.

System Action: The program ends.

Administrator Response: Ensure that the input file exists and try the operation again.

ASUV00709I Blank line is missing between two identity mapping entries.

Severity: svc_c_sev_warning

Explanation: A blank line is needed to separate identity mapping entries.

System Action: Processing of the identity mapping file continues.

Administrator Response: Ensure that a blank line exists between identity mappings. Then try the request again.

ASUV00710I Blank line detected in identity mapping file entry.

Severity: svc_c_sev_warning

Explanation: A blank line is detected in the identity mapping file that is not used to separate entries.

System Action: Processing of the identity mapping file continues.

Administrator Response: Remove any unnecessary blank lines from the identity mapping file.

ASUV00711A Environment variable _ASU_IDMAP is not specified correctly or is missing.

Severity: svc_c_sev_error

Explanation: The environment variable _ASU_IDMAP is not specified correctly or is missing from the Application Support server environment variable file.

System Action: The program ends.

Administrator Response: Ensure that the environment variable exists and is pointing to an identity mapping file. Try the operation again.

ASUV00718I Access control list database configuration is successful.

Severity: svc_c_sev_notice

Explanation: Application Support server access control list database configuration is successful.

System Action: The program continues.

ASUV00719A DCE return code is *status_code*.

Severity: svc_c_sev_error

Explanation: A call to a DCE service has failed. The return code *status_code* explains the failure.

System Action: The program continues with reduced function.

Administrator Response: Correct the error by using the DCE status code response information found in *z/OS DCE Messages and Codes*, SC24-5912. Try the request again. If the problem continues, contact the service representative.

ASUV00720A No RACF mapping exists for the specified UUID.

Severity: svc_c_sev_error

Explanation: The mapping for the specified UUID does not exist in RACF.

System Action: The program continues.

Administrator Response: Add the user making the request to RACF.

ASUV00721A The external security manager is not available or is at the wrong level.

Severity: svc_c_sev_error

Explanation: The external security manager is at the wrong level or is not available and the environment variable `_ASU_EX_SEC` is YES.

System Action: The program continues.

Administrator Response: Verify the external security manager is available and is at an appropriate level. Refer to Mapping DCE Principal Names to MVS User IDs with RACF in *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903.

ASUV00722A The RACF DCEUIDS class is not active.

Severity: svc_c_sev_error

Explanation: The RACF DCEUIDS class must be active in order for the Application Support server to obtain the MVS userid.

System Action: The program continues.

Administrator Response: Add the user making the request to RACF.

ASUV00723A Error in RACF processing. Call to IRRSUD00 failed. SAF return code = *SAF_return_code*, RACF return code = *RACF_return_code*, RACF reason code = *RACF_reason_code*.

Severity: svc_c_sev_error

Explanation: The identity mapping function in RACF failed.

System Action: The program continues.

ASUV00724I Reload option ignored - not valid with RACF identity mapping.

Severity: svc_c_sev_notice

Explanation: The reload option is not supported when RACF is used for getting the MVS ID.

System Action: The program continues.

ASUV00725A The Application Support server is not authorized to use RACF for identity mapping.

Severity: svc_c_sev_error

Explanation: The Application Support server is not authorized to use RACF to do the identity mapping function and `ASU_EX_SEC` is set to YES.

System Action: The program continues.

Administrator Response: In order to use RACF for identity mapping the RACF FACILITY class must have the IRR.RDCERUID profile defined and the Application Support Server needs READ access.

ASUV00726I The Application Support server identity mapping table was created but is not complete due to errors in the identity mapping file.

Severity: svc_c_sev_warning

Explanation: The identity mapping table used by the Application Support server was created. It may not be complete because of errors found during identity mapping file processing.

System Action: Server initialization or RELOAD continues.

Administrator Response: Verify that the errors in the identity mapping file are acceptable. If not, correct the errors and reload the file. For more information on the Application Support identity mapping function, refer to *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903.

ASUV00727A Application Support server can not initialize its internal recovery component.

Severity: svc_c_sev_error

Explanation: Initialization can not continue because the Application Support server can not initialize the internal recovery component.

System Action: Application Support server stops initialization.

Administrator Response: If the problem continues, contact the service representative.

ASUV00728A Internal error in the Application Support server recovery component: *error_information*.

Severity: svc_c_sev_error

Explanation: Internal error with the Application Support server recovery component. *error_information* is a data string given for use by Service personnel.

System Action: Application Support server stops.

Administrator Response: If the problem continues, contact the service representative.

ASUV00729I OS/390 Recoverable Resource Management Services has lost hardened data for resource manager *resource_manager*.

Severity: svc_c_sev_notice

Explanation: Recoverable Resource Management Services has lost hardened data and may not be able to return data for all incomplete URs for this *resource_manager*.

System Action: Application Support server continues.

Administrator Response: Using RRS message ATR212I to determine the range of affected transactions, try to resolve any unresolved in-doubt transactions. If the problem continues, contact the service representative.

ASUV00730A OS/390 Recoverable Resource Management Services is not started.

Severity: svc_c_sev_error

Explanation: Recoverable Resource Management Services has not been started.

System Action: Application Support server stops.

Administrator Response: Start Recoverable Resource Management Services.

ASUV00731I Distributed transactional processing is enabled.

Severity: svc_c_sev_notice

Explanation: The Application Support server can accept transactional RPCs.

System Action: Application Support server continues.

Administrator Response: None.

ASUV00732I Distributed transactional processing is not enabled.

Severity: svc_c_sev_notice

Explanation: The Application Support server can not accept transactional RPCs. The environment variable `_ASU_TRAN` is set to NO.

System Action: Application Support server continues.

Administrator Response: None.

ASUV00733A The Application Support server is not registered with the OS/390 Recoverable Resource Management Services.**Severity:** svc_c_sev_error**Explanation:** The Recoverable Resource Management Services have reset the resource manager status associated with the Application Support server. This could happen if RRS was stopped and restarted while the Application Support server was active.**System Action:** Application Support server stops.**Administrator Response:** Restart the Application Support server.**ASUV00734A Application Support failure type *error_type*, reason *error_reason* ('*error_reason*'x) during message queue initialization.****Severity:** svc_c_sev_error**Explanation:** The Application Support server cannot create the message queues and processing threads required to communicate with IMS Message Processing Regions. The failing system service or error is identified by *error_type*. The code from the failing service is placed in *error_reason*, if one is applicable.**System Action:** Application Support server stops.**Administrator Response:** Record *error_type*, and refer to the table below for the appropriate response for that error.

Error	Response
1	Modify your Application Support Server job to specify an <code>_EUV_HOME</code> variable. <i>error_reason</i> does not apply.
2	There was insufficient storage to create queues. <i>error_reason</i> does not apply. Resubmit the job. If the problem continues, contact the service representative.
3	An error occurred during <code>pthread_create()</code> . If the problem continues, contact the service representative.
4	An error occurred during <code>ftok()</code> . If the problem continues, contact the service representative.
5	An error occurred during <code>msgctl()</code> . If the problem continues, contact the service representative.
6	An error occurred during <code>msgget()</code> . If the problem continues, contact the service representative.

ASUV00735A Application Support encountered error '*error_reason*' during an '*ipc_operation*' operation.**Severity:** svc_c_sev_error**Explanation:** The Application Support server encountered an unexpected error during an interprocess communication '`msgrcv`' or '`msgsnd`' operation.**System Action:** Application Support server stops. All Encina transactions this server is processing are aborted.**Administrator Response:** If the problem continues, contact the service representative.

ASUV00736I Application Support encountered an IPC request that is not valid.

Severity: svc_c_sev_notice

Explanation: The Application Support server received an IPC message in a format that is not valid.

System Action: Application Support server continues. The Encina transaction that initiated the IPC request is aborted.

Administrator Response: If the problem continues, contact the service representative.

ASUV00750I Application Support identity mapping file reload is successful.

Severity: svc_c_sev_notice

Explanation: The identity mapping file reload function is successful.

System Action: The program continues.

ASUV00800I MODIFY command is missing parameters.

Severity: svc_c_sev_warning

Explanation: The Application Support server receives a MODIFY command that has no parameters or is missing parameters. The command is ignored.

System Action: The Application Support server ignores the command and continues.

Operator Response: Ensure that the command syntax is correct, then try the request again.

ASUV00801E MODIFY command is too long.

Severity: svc_c_sev_warning

Explanation: The MODIFY command is too long.

System Action: The Application Support server ignores the command and continues.

Operator Response: Do one of the following, then try the request again:

- Use abbreviations for command parameters to reduce the length of the command.
 - Ensure that the command syntax is correct.
-

ASUV00802A Application Support server cannot retrieve MODIFY command data.

Severity: svc_c_sev_error

Explanation: The Application Support server encounters a severe error while attempting to retrieve the MODIFY command data from the communications area for the task.

System Action: The program writes a dump and continues with reduced function. MODIFY commands are no longer processed.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00803A MODIFY command contains no verb.

Severity: svc_c_sev_error

Explanation: The verb parameter is not specified on the MODIFY command line. The parameter is required.

System Action: The program continues.

Operator Response: Ensure that the command syntax is correct, then try the request again.

ASUV00804A Application Support server does not recognize the verb *verb_name*.

Severity: svc_c_sev_error

Explanation: The verb parameter supplied on the MODIFY command line is not valid.

System Action: The program continues.

Operator Response: Ensure that the command syntax is correct, then try the request again. A valid command must be one of

- /MODIFY <jobname>,RELOAD
- /MODIFY <jobname>,STARTATT
- /MODIFY <jobname>,STOPATT
- /MODIFY <jobname>,DISPSRVR
- /MODIFY <jobname>,DISPIF
- /MODIFY <jobname>,INSTIF
- /MODIFY <jobname>,UNINSTIF

ASUV00805A Application Support server cannot allocate storage.

Severity: svc_c_sev_error

Explanation: The Application Support server cannot allocate storage.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00806E Application Support server cannot create a thread. *error_text*

Severity: svc_c_sev_error

Explanation: The Application Support server attempts to create a thread. The attempt fails. Additional information about the error is given by the C library *error_text*, which is returned from the **pthread_create()** routine. In most cases, the reason for the failure is: that the system limit for the maximum number of threads per process has been reached.

System Action: The program writes a dump and continues.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Do one of the following, then try the request again:

- Wait for at least one request in progress to complete.
- Ensure that the MAXTHREADTASKS and MAXTHREADS values in the z/OS parmlib member are set to sufficient levels to accommodate the system load.

Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00807E Application Support server cannot detach a thread. *error_text*

Severity: svc_c_sev_error

Explanation: The Application Support server cannot detach a thread. Additional information about the error is given by the C library *error_text*, which is returned from the **pthread_detach()** routine.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00808A Application Support server detects an exception. The request *request_name* fails.

Severity: svc_c_sev_error

Explanation: The Application Support server detects an exception while processing a request *request_name*. This request could not complete, but no services will be terminated.

System Action: The program writes a dump and continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00809A Application Support server detects an exception. Ending service *service_name*.

Severity: svc_c_sev_error

Explanation: The Application Support server detects an exception. The service that detects the exception will no longer be available.

System Action: The program writes a dump and continues with reduced function. The service that detects the exception ends.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: The service *service_name* ends. If you do not want the Application Support server to continue without the service then CANCEL the Application Support server and START it again. Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00810A Application Support server cannot initialize MODIFY queue.

Severity: svc_c_sev_fatal

Explanation: Application Support server cannot initialize the MODIFY command queue.

System Action: The program writes a dump and continues with reduced function. The MODIFY command service is not available.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00811E Application Support server Stop request rejected.

Severity: svc_c_sev_error

Explanation: The Application Support server is not in a state that allows the server to stop.

System Action: The program continues.

Operator Response: Wait for completion of any outstanding administrative commands. When the Application Support server reaches operational or quiescent state, try the Stop request again.

ASUV00812I Application Support server stops.

Severity: svc_c_sev_notice

Explanation: The Application Support server process ends.

System Action: The MVS job or started task for the Application Support server is ended.

ASUV00813I Application Support server operator request fails.

Severity: svc_c_sev_error

Explanation: The request from the operator console cannot be processed.

System Action: The program continues.

Operator Response: See any other error messages generated by the Application Support server as a result of the operator request. Correct the error and try the command again. If no other error messages are produced, issue the Application Support server command from the administrative utility instead of the operator console.

ASUV00814E Application Support server cannot synchronize a thread.

Severity: svc_c_sev_error

Explanation: The Application Support server attempts to join a console executor thread. The attempt fails.

System Action: The program continues.

Operator Response: Try the request again.

ASUV00815I *server_data*

Severity: svc_c_sev_notice

Explanation: The operator issued a MODIFY command with the DISPSRVR verb parameter. A multi-line message with server data in positional fields separated by commas is displayed on the operator console as follows:

Line one contains up to 55 characters of the Application Support server CDS name.

Line two contains the Application Support server subsystem type (IMS OTMA, IMS APPC, IMS ISC, CICS), state, start time, and level. The server state is represented as a number as follows:

- 1 = Initializing
- 2 = Quiescent
- 3 = Starting
- 4 = Operational
- 5 = Stopping
- 6 = Terminating

The server level is the genned date and time.

Line three contains the time the attachment facility was started or stopped, the MVS ID of the administrator who started or stopped the attachment facility (this will be MVSOP if the start or stop was issued from the operator console), a message number whose associated text states why the attachment facility is in its current state, the number of installed interfaces, and the number of available RPC threads.

For CICS and IMS ISC Application Support servers, there is a fourth line containing additional server information. For CICS servers, this consists of the APPLID of the attached CICS system and the APPLID of the Application Support server. For IMS ISC servers, line four contains the IMS LU name, the logon mode name, the Application Support server LU name, and the device prefix.

System Action: The program continues.

ASUV00816I *interface_data*

Severity: svc_c_sev_notice

Explanation: The operator issued a MODIFY command with the DISPIF verb parameter and interface name specified. A message with interface information in positional fields separated by commas is displayed on the operator console.

The interface name is in the first positional field. The second field contains a number representing the interface state as follows:

- 0 = Interface is not installed
- 1 = Interface is installed
- 2 = Interface is installed but not in STUB library
- 3 = Interface is draining

An interface that is in the installed state will also have the major version number and minor version number displayed with a period between them, followed by a number representing the interface type:

- 1 = CICS
- 2 = IMS

If the interface is transactional, an asterisk will immediately follow the interface type number.

The last field in the message contains the number of operations in the interface definition.

An interface that is in any state other than installed will have only the interface name and state displayed.

System Action: The program continues.

ASUV00817I *interface_list*

Severity: svc_c_sev_notice

Explanation: The operator issued a MODIFY command with the DISPIF verb parameter and no interface name specified. Message ASUV00817I will be displayed on the console as many times as necessary to display the names of all installed interfaces. Up to six interface names, separated by blanks, will be displayed in each message.

System Action: The program continues.

ASUV00900A **Parameters for ASUVIFM program are not correct.**

Severity: svc_c_sev_error

Explanation: The Application Support bulk install program (ASUVIFM) is not called correctly. The syntax is:

ASUVIFM parm1 parm2

where parm1 is the operation (1 for install, 0 for uninstall).

parm2 is the Application Support server name.

System Action: The program ends.

Administrator Response: Ensure that the command syntax is correct, then try the request again.

ASUV00901A Application Support bulk install program cannot bind to Application Support server. DCE status code: *status_code*.

Severity: svc_c_sev_error

Explanation: The Application Support bulk install program cannot retrieve a binding handle to the Application Support server. The DCE status code is *status_code*.

System Action: The program ends.

Administrator Response: Refer to the *z/OS DCE Messages and Codes*, SC24-5912, to look up the return code from RPC. Follow the action specified and try the Application Support bulk install program again.

ASUV00902I Action request for interface *interface* is successful.

Severity: svc_c_sev_notice

Explanation: The action request to install or uninstall an interface is successful.

System Action: The program continues.

ASUV00903A Load module for interface *interface* is not found in Application Support server stub library.

Severity: svc_c_sev_error

Explanation: In most cases, the reasons for the failure are:

- The interface name is not spelled correctly.
- The interface is missing from the Application Support server stub library.
- The stub library is not APF-authorized and the AS IMS server module runs from an APF-authorized library.

System Action: The program continues.

Administrator Response: Ensure that the interface name is spelled correctly and is specified in the Application Support server stub library, and try the Application Support bulk install program or INSTIF command again. If the problem continues, contact the service representative.

ASUV00904A Cannot export interface *interface* to the directory.

Severity: svc_c_sev_error

Explanation: Interface *interface* cannot be exported to the CDS directory.

System Action: The program continues.

Administrator Response: Ensure that the directory service is available and try the Application Support bulk install program or INSTIF command again. If the problem continues, contact the service representative.

ASUV00905A Cannot register interface *interface* with local endpoint map.

Severity: svc_c_sev_error

Explanation: The Application Support bulk install program or MODIFY command cannot register interface *interface* with the local endpoint map.

System Action: The program continues.

Administrator Response: Check the Application Support server message log for messages. Ensure that the DCE Host daemon is available and try the Application Support bulk install program or INSTIF command again. If the problem continues, contact the service representative.

ASUV00906A Cannot register interface *interface* with RPC run time.

Severity: svc_c_sev_error

Explanation: The Application Support bulk install program or INSTIF command cannot register the interface *interface* with the RPC run time.

System Action: The program continues.

Administrator Response: Check the Application Support server message log for messages. Ensure that the RPC function is available and try the Application Support bulk install program or INSTIF command again. If the problem continues, contact the service representative.

ASUV00907A Error detected while trying to load interface *interface*.

Severity: svc_c_sev_error

Explanation: The Application Support bulk install program of MODIFY command detects an error while trying to load interface *interface*.

System Action: The program continues.

Administrator Response: Ensure that the interface is in the Application Support server stub library. Try the Application Support bulk install program or INSTIF command again. If the problem continues, contact the service representative.

ASUV00908I Interface *interface* is already installed.

Severity: svc_c_sev_notice

Explanation: The Application Support bulk install program or INSTIF command detects that the interface *interface* is already installed.

System Action: The program continues.

Administrator Response: The duplicate interface should be removed from SYSIN for the Application Support bulk install program.

ASUV00909A Maximum number of interfaces allowable is exceeded while installing *interface*.

Severity: svc_c_sev_error

Explanation: The Application Support bulk install program or INSTIF command attempts to install more than 500 interfaces.

System Action: The program continues.

Administrator Response: The excess interfaces should be removed from SYSIN for the Application Support bulk install program.

ASUV00910I Cannot uninstall interface *interface* because interface is not installed.

Severity: svc_c_sev_notice

Explanation: The Application Support bulk install program or UNINSTIF command attempts to uninstall the interface *interface*, but it is not installed.

System Action: The program continues.

ASUV00911A Interface *interface* is installed but not in Application Support server stub library.

Severity: svc_c_sev_error

Explanation: The Application Support server or UNINSTIF command cannot uninstall the interface because the interface is not in the Application Support server stub library. In most cases, the reason for the failure is: that the stub library may have been replaced since the interface was installed.

System Action: The program continues.

Administrator Response: Ensure that the Application Support server stub library contains the installed interface, and try the Application Support bulk install program or UNINSTIF command again.

ASUV00912I Interface *interface* uninstall is pending.

Severity: svc_c_sev_notice

Explanation: The uninstall request for interface *interface* is waiting for other transactions using the interface *interface* to complete. The interface *interface* will be uninstalled when the transactions are complete.

System Action: The program continues.

ASUV00913A Interface *interface* is a duplicate of interface *interface_2*.

Severity: svc_c_sev_error

Explanation: Interface *interface* and interface *interface_2* have different names, but they are identical interfaces.

System Action: The program continues. Interface *interface_2* is not installed.

Administrator Response: One of the two interfaces should be removed from SYSIN for the Application Support bulk install program.

ASUV00914A Unknown response received from the Application Support server while processing interface *interface*.

Severity: svc_c_sev_error

Explanation: The Application Support server returns a response which is not known to the Application Support bulk install program or INSTIF command while trying to install the interface *interface*. The resulting state of the interface is not known.

System Action: The program continues.

Administrator Response: Additional information or messages may be found in the Application Support server message log. If the problem continues, contact the service representative.

ASUV00915A Operation *operation* is not valid as input to Application Support bulk install program.

Severity: svc_c_sev_error

Explanation: The interface management operation *operation* is not a valid input parameter to the Application Support bulk install program.

System Action: The program ends.

Administrator Response: Ensure the input parameter is valid for the Application Support bulk install program. Refer to the *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903, for the correct syntax and try the request again.

ASUV00916A Application Support bulk install program detects the number of interfaces specified in SYSIN exceeds the maximum.

Severity: svc_c_sev_error

Explanation: The Application Support bulk install program will only install or uninstall 500 interfaces. The list of interfaces specified in SYSIN exceeds the maximum.

System Action: The program ends.

Administrator Response: Ensure that the number of interfaces specified in SYSIN is less than or equal to 500 interfaces and try the Application Support bulk install program again.

ASUV00917I Application Support bulk install program only allows a maximum of *number* interfaces.

Severity: svc_c_sev_notice

Explanation: *number* is the maximum interfaces allowed by the Application Support bulk install program. The number of interfaces specified in SYSIN exceeds the maximum.

System Action: The program ends.

Administrator Response: Ensure that the number of interfaces specified in SYSIN for the Application Support bulk install program does not exceed the maximum. Try the Application Support bulk install program again.

ASUV00918A Application Support bulk install program detects error that is not expected while processing standard input.

Severity: svc_c_sev_error

Explanation: The Application Support bulk install program detects an error that is not expected while processing the list of interfaces.

System Action: The program ends.

Administrator Response: Check the format of the input to the Application Support bulk install program. Try the Application Support bulk install program again. If the problem continues, contact the service representative.

ASUV00919A Interface name *interface* is too long.

Severity: svc_c_sev_error

Explanation: The interface name *interface* provided in SYSIN or via the DISPIF, INSTIF, or UNINSTIF commands is more than 8 characters.

System Action: The interface *interface* will not be processed, but the Application Support bulk install program will continue processing any remaining interfaces.

Administrator Response: Ensure that the interface name is a maximum of 8 characters and try the request again.

ASUV00920A Application Support bulk install program detects that Application Support server name input parameter is too long.

Severity: svc_c_sev_error

Explanation: The Application Support server name that is an input parameter to the Application Support bulk install program is longer than 256 characters.

System Action: The program ends.

Administrator Response: Correct the Application Support server name and try the Application Support bulk install program again.

ASUV00921I Application Support bulk install program is successful in installing *number* interfaces.

Severity: svc_c_sev_notice

Explanation: The Application Support bulk install program is successful in installing *number* interfaces.

System Action: The program continues.

ASUV00922I Application Support bulk install program cannot install *number* interfaces.

Severity: svc_c_sev_notice

Explanation: The Application Support bulk install program cannot install *number* interfaces. Each interface that cannot be installed is listed in standard error with an explanation of the cause of the failure.

System Action: The program continues.

ASUV00923I Application Support bulk install program cannot uninstall specified interfaces.

Severity: svc_c_sev_notice

Explanation: The Application Support bulk install program cannot uninstall the specified interfaces. Additional information about the error is logged by the Application Support server.

System Action: The program continues.

ASUV00924I Application Support server is using RPC default for number of executor threads.

Severity: svc_c_sev_notice

Explanation: The Application Support server cannot determine a meaningful number of RPC executor threads to start, so it is using the RPC default. In most cases, the reasons for the failure are:

- For CICS, the environment variable `_ASU_CICS_CONNECTIONS` for CICS connections is not specified or is not accessible.
- For IMS, this message indicates an internal failure.

System Action: The Application Support server continues to initialize and will be usable, although it may run with reduced function because it uses a number of executor threads that is smaller or larger than the number the administrator may wish to use.

ASUV00925I Application Support bulk install program uninstalled *number* interfaces successfully.

Severity: svc_c_sev_notice

Explanation: The Application Support bulk install program is successful in uninstalling *number* interfaces.

System Action: The program continues.

ASUV00926I Application Support bulk install program cannot uninstall *number* interfaces.

Severity: svc_c_sev_notice

Explanation: The Application Support bulk install program cannot uninstall *number* interfaces successfully. Each interface that is not uninstalled is listed in standard error with an explanation on the cause of the failure.

System Action: The program continues.

ASUV00927A Application Support server cannot open log file ddname ASUVPLOG.

Severity: svc_c_sev_error

Explanation: The Application Support server cannot open the log file specified by the DD statement ASUVPLOG.

System Action: The program continues with reduced function.

System Programmer Response: Correct the DD statement for the ASUVPLOG dataset for the Application Support server. The dataset must be sequential. If the problem continues, contact the service representative.

ASUV00928A Application Support CICS cannot set thread specific data. *error_text*

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility is not able to set thread specific data. Information about the error is given by the C library *error_text*, which is returned from the **pthread_setspecific()** function.

Additional diagnostic information is available from the dump and probe of user abend xBBF.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00929A Application Support CICS cannot get thread specific data. *error_text*

Severity: svc_c_sev_error

Explanation: The Application Support CICS Attach Facility is not able to get thread specific data. Information about the error is given by the C library *error_text*, which is returned from the **pthread_getspecific()** function.

Additional diagnostic information is available from the dump and probe of user abend xBBF.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00930A Cannot initialize ISPF. Call to fetch ISPF load module ISPLINK fails.

Severity: svc_c_sev_fatal

Explanation: The Application Support administration program cannot dynamically load the ISPF module ISPLINK. The call to the C library function **fetch()** fails.

System Action: The program ends.

System Programmer Response: Ensure that the user ID invoking the Application Support administration program has the module ISPLINK in the steplib, and try the request again. If the problem continues, contact the service representative.

ASUV00931A Application Support identity mapping function cannot allocate storage..

Severity: svc_c_sev_fatal

Explanation: The Application Support server identity mapping function (either at system initialization time or during the reload operation) cannot allocate storage necessary for command processing. This error should not occur during normal processing.

System Action: The program writes a dump and ends abnormally.

Administrator Response: Increase the region size for the Application Support server and try the request again. Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00932A Application Support server cannot allocate storage for *string*.

Severity: svc_c_sev_error

Explanation: Application Support server cannot allocate storage for *string*. This error should not occur during normal processing.

System Action: The program continues with reduced function.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Increase the region size for the Application Support server. Try the request again. Use the dump to help determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00933A CICS call fails and returns response code *response_code*, reason code *reason_code*, subreason field_1 *subreason_field1*, subreason field_2 *subreason_field2* and message pointer *msg_ptr*.

Severity: svc_c_sev_error

Explanation: A call to one of the external CICS (EXCI) interfaces fails. The failing EXCI interface is identified by a previous message. Additional information is found in the *CICS External Interfaces Guide*, SC34-5709.

System Action: The Application Support server writes a dump and stops the Attach Facility.

Operator Response: Collect the dump and probe information and notify the system programmer.

System Programmer Response: Use the CICS information and dump to determine the probable cause of the failure. If the problem continues, save the dump and contact the service representative.

Additional information about dumps is found in the *z/OS Language Environment Debugging Guide*, GA22-7560.

ASUV00934A Application Support server errors exceed threshold limit. Application Support server stops Attach Facility.

Severity: svc_c_sev_error

Explanation: The Application Support server encounters a number of errors which exceed the threshold limit of the server. A previous message is issued that provides additional information about the error. The Attach Facility is stopped.

System Action: The program continues with reduced function.

Administrator Response: Check the console log for messages that occur prior to this message to determine the reason for failure, and correct the error based on the message text. Then start the attachment again. If the problem continues, contact the service representative.

ASUV00935A Application Support server errors exceed threshold limit. Application Support server is ending.

Severity: svc_c_sev_fatal

Explanation: The Application Support server encounters a number of errors which exceed the threshold limit of the server. A previous message is issued that provides additional information about the error. The Application Support server is stopped.

System Action: The program ends.

Operator Response: Check the console log for any messages, probes, or dumps that occur prior to the termination message, and notify the system programmer.

System Programmer Response: Correct the error based on the previous message text, and start the Application Support server again. If the problem continues, contact the service representative.

ASUV00936A Application Support server cannot obtain system time.

Severity: svc_c_sev_fatal

Explanation: The Application Support server cannot obtain the system time from the DCE kernel.

System Action: The program ends.

Operator Response: Ensure that the DCE kernel is active and try the request again. If the problem continues, contact the service representative.

Appendix H. Codes

This section lists abend codes and reason codes.

Abend Codes From DCE AS CICS

This is the list of abend codes from Application Support/CICS.

BBF

Explanation: The Attach Facility finds an error in a CICS Application Support server address space. The error is indicated by the hexadecimal *reason code* in register 15. The reason codes are as follows:

Reason Explanation

- | | |
|-----------|--|
| 01 | Error occurs while allocating a pipe to a target CICS system. |
| 02 | Error occurs while opening a pipe from the Application Support server to the attached CICS system. |
| 03 | Error occurs while communicating a request for program execution from the Application Support server to the attached CICS system. |
| 04 | Error occurs while closing a pipe between the Application Support server and the attached CICS system. |
| 05 | Error occurs while deallocating a pipe between the Application Support server and the attached CICS system. |
| 06 | An unexpected error occurs. Entry to the recovery routine can be caused by a program or machine check interrupt or an unsolicited ABEND. |
| 07 | An unexpected condition in the Attach Facility is recognized. This can be caused by damage to the control block structures required for continued Attach Facility operation. |
| 08 | The Attach Facility cannot establish an end-of-task resource manager for one or more RPC executor threads. |
| 09 | Error occurs while initializing the CICS Attach Facility environment required for continued operation. |
| 0A | Error occurs while obtaining storage during the Start Attach Facility processing. |

System Action: Attach facility recovery routines are entered, a system SVC dump is requested, and other diagnostic information is recorded in SYS1.LOGREC. Typically the recovery routines try to mainline Attach Facility logic to cause an internal stop of the Attach Facility.

System Programmer Response: Take the appropriate action as indicated by the messages produced and the Attach Facility state reason associated with the problem. Also do the following:

- Use the EREP service aid to dump the SYS1.LOGREC data set and save the output.
- Save the SVC dump.
- Try to use the ASUADMIN program to determine the state of the Attach Facility and the associated state reason.

Then try the request again.

Server State Reason Codes

There are three types of server state reason codes:

- Application Support server codes which apply to both DCE AS CICS and DCE AS IMS
- DCE AS CICS codes
- DCE AS IMS codes

Application Support Server State Reason Codes

Application Support server state reason codes which apply to both DCE AS CICS and DCE AS IMS have negative values and are summarized in the following table:

Code	Meaning
-1	Application Support server has been initialized.
-2	The administrator has issued a Start Attachment command.
-3	The administrator has issued a Stop Attachment command.
-4	Application Support server has been stopped because of internal server error.
-5	Attachment Facility has been stopped because of DCED error.
-6	Attachment Facility has been stopped because of Directory failure.
-7	Attachment Facility has been stopped because of DCE Runtime failure.

DCE AS CICS State Reason Codes

The following table summarizes the state reason codes DCE AS CICS sets:

Code	Meaning
1	Storage allocation error
2	Initialization failure
3	Insufficient number of sessions
4	End of transmission was not established.
5	Logic error
6	No CICS
7	TCB initialization failure
8	Unable to allocate a pipe
9	Unknown CICS error
10	No IRC was started
11	DFHXXXXX load failed
12	Initialization user error
13	Initialization user system error
14	SSI verification failed
15	Open pipe system error
16	Allocation pipe system error
17	IRC logon failure
18	No pipe available
19	IRC connection failure
20	Allocation pipe user error
21	Open pipe user error
22	Close pipe user error
23	IRC disconnect failure

Code	Meaning
24	Close pipe system error
25	Deallocation pipe user error
26	IRC logoff failure
27	Deallocation pipe system error
28	DPL user error
29	DPL system error
30	Version number is not valid
31	CICS release is not valid
32	Connection definition is not valid
33	Unable to create ESTAE
34	Unexpected error
35	Cannot build the hash table
36	Corrupt binary file
37	Cannot access the ID mapping file

DCE AS IMS State Reason Codes

The following table summarizes the state reason codes set by DCE AS IMS:

Code	Meaning
100	Fatal internal error
101	MUTEX error
102	Fatal memory error
103	Fatal VTAM error
104	Password is not valid
105	No VTAM
106	Subsystem ID is not valid
107	Logwriter open failed

Appendix I. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

AIX	BookManager	CICS
CICS/ESA	IBM	IBMLink
IMS	IMS/ESA	Language Environment
Library Reader	OS/2	OS/390
RACF	Resource Link	SecureWay
System/370	z/OS	

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Programming Interface Information

This *z/OS DCE Application Support Configuration and Administration Guide* documents intended Programming Interfaces that allow the customer to write programs to obtain services of Application Support.

Glossary

This glossary defines technical terms and abbreviations used in z/OS DCE documentation. If you do not find the term you are looking for, refer to the index of the appropriate z/OS DCE manual or view the *IBM Glossary of Computing Terms*, located at:

<http://www.ibm.com/ibm/terminology>

This glossary includes terms and definitions from:

- *IBM Dictionary of Computing*, SC20-1699.
- *Information Technology—Portable Operating System Interface (POSIX)*, from the POSIX series of standards for applications and user interfaces to open systems, copyrighted by the Institute of Electrical and Electronics Engineers (IEEE).
- *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1.SC1).
- *CCITT Sixth Plenary Assembly Orange Book, Terms and Definitions* and working documents published by the International Telecommunication Union, Geneva, 1978.
- Open Software Foundation (OSF).

The following abbreviations indicate terms that are related to a particular DCE service:

CDS	Cell Directory Service
CICS/ESA®	Customer Information Control System/ESA
DTS	Distributed Time Service
GDS	Global Directory Service
IMS/ESA®	Information Management System/ESA
RPC	Remote Procedure Call
Security	Security Service
Threads	Threads Service
XDS	X/Open Directory Services
XOM	X/Open OSI-Abstract-Data Manipulation

A

abort. To fail to commit. Any changes made by an Encina transaction that is aborted, for whatever reason, are undone. Once an Encina transaction is undone (rolled back), no evidence that it was ever attempted remains outside of records in the transaction processing system's log. See also *commit*.

access control list (ACL). (1) GDS: Specifies the users with their access rights to an object. (2) Security: Data that controls access to a protected object. An ACL specifies the privilege attributes needed to access the object and the permissions that may be granted, to the protected object, to principals that possess such privilege attributes.

access right. Synonym for *permission*.

accessible. Pertaining to an object whose client possesses a valid designator or handle.

account. Data in the Registry database that allows a principal to log in. An account is a registry object that relates to a principal.

ACL. Access control list.

adapter. Synonym for *attachment facility*.

address. An unambiguous name, label, or number that identifies the location of a particular entity or service. See *presentation address*.

Advanced Program-to-Program Communications (APPC). An implementation of the SNA/SDLC LU6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

APF. Authorized program facility.

API. Application program interface.

APPC. Advanced Program-to-Program Communications.

application identifier. A unique identifier used to identify an application in the transactional RPCs sent in a distributed environment.

application program interface (API). A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to

use specific data or functions of the operating system or the licensed program.

Application Support server. Refers to the server for z/OS DCE Application Support. The Application Support server allows a client program to access CICS or IMS.

application thread. A thread of execution created and managed by application code. See *client application thread*, *local application thread*, *RPC thread*, and *server application thread*.

applid. Application identifier. A unique identifier used to identify an application in the transactional RPCs sent in a distributed environment.

architecture. (1) The organizational structure of a computer system, including the interrelationships among its hardware and software. (2) The logical structure and operating principles of a computer network. The operating principles of a network include those of services, functions, and protocols.

ASPREFX. The high-level qualifier of the MVS data set where your Application Support files are installed.

asynchronous. Without a regular time relationship; unexpected or unpredictable with respect to the running of program instructions.

attachment facility. Application Support server: Refers to the CICS adapter and the IMS adapter. Synonymous with *adapter*.

attribute. (1) RPC: An Interface Definition Language (IDL) or attribute configuration file (ACF) that conveys information about an interface, type, field, parameter, or operation. (2) DTS: A qualifier used with DTS commands. DTS has four attribute categories: characteristics, counters, identifiers, and status. (3) XDS: Information of a particular type concerning an object and appearing in an entry that describes the object in the directory information base (DIB). It denotes the attribute's type and a sequence of one or more attribute values, each accompanied by an integer denoting the value's syntax.

attribute syntax. GDS: A definition of the set of values that an attribute may assume. Attribute syntax includes the data type, in ASN.1, and usually one or more matching rules by which values may be compared.

attribute type. (1) XDS: The component of an attribute that indicates the type of information given by that attribute. Because it is an object identifier, it is unique among other attribute types. (2) XOM: Any of various categories into which the client dynamically groups values on the basis of their semantics. It is an integer unique only within the package.

attribute value. XDS, XOM: A particular instance of the type of information indicated by an attribute type.

authentication. In computer security, a method used to verify the identity of a principal.

Authentication Service. One of three services provided by the Security Service: it verifies principals according to a specified authentication protocol. The other Security services are the Privilege Service and the Registry Service.

authorization. (1) The determination of a principal's permissions with respect to a protected object. (2) The approval of a permission sought by a principal with respect to a protected object.

authorized program facility (APF). An MVS facility that permits identification of programs authorized to use restricted functions.

B

basic mapping support (BMS). A facility that moves data streams to and from a terminal in CICS. It is an interface between CICS and its application programs. It formats input and output display data in response to BMS commands in programs.

binding. RPC: A relationship between a client and a server involved in a remote procedure call.

binding handle. RPC: A reference to a binding. See *binding information*.

binding information. RPC: Information about one or more potential bindings, including an RPC protocol sequence, a network address, an endpoint, at least one transfer syntax, and an RPC protocol version number. See *binding*. See also *endpoint*, *network address*, *RPC protocol*, *RPC protocol sequence*, and *transfer syntax*.

BMS. Basic mapping support.

broadcast. A notification sent to all members within an arbitrary grouping such as nodes in a network or threads in a process. See also *signal*.

C

cache. (1) CDS: The information that a CDS clerk stores locally to optimize name lookups. The cache contains attribute values resulting from previous lookups, as well as information about other clearinghouses and namespaces. (2) Security: Contains the credentials of a principal after the DCE login. (3) GDS: See *DUA cache*.

call thread. RPC: A thread created by an RPC server's runtime to run remote procedures. When engaged by a remote procedure call, a call thread temporarily forms part of the RPC thread of the call. See *application thread* and *RPC thread*.

cancel. (1) Threads: A mechanism by which a thread informs either itself or another thread to stop the thread as soon as possible. If a cancel arrives during an important operation, the canceled thread may continue until it can end the thread in a controlled manner. (2) RPC: A mechanism by which a client thread notifies a server thread (the canceled thread) to end the thread as soon as possible. See also *thread*.

CDS. Cell Directory Service.

CDS clerk. The software that provides an interface between client applications and CDS servers.

CDS control program (CDSCP). A command interface that CDS administrators use to control CDS servers and clerks and manage the name space and its contents. See also *manager*.

CDSCP. CDS control program.

cell. The basic unit of operation in the distributed computing environment. A cell is a group of users, systems, and resources that are grouped around a common purpose and that share common DCE services.

Cell Directory Service (CDS). A DCE component. A distributed replicated database service that stores names and attributes of resources located in a cell. CDS manages a database of information about the resources in a group of machines called a DCE cell.

CICS. Customer Information Control System.

class. A category into which objects are placed on the basis of their purpose and internal structure.

clerk. (1) DTS: A software component that synchronizes the clock for its client system by requesting time values from servers, calculating a new time from the values, and supplying the computed time to client applications. (2) CDS: A software component that receives CDS requests from a client application, ascertains an appropriate CDS server to process the requests, and returns the results of the requests to the client application.

client. A computer or process that accesses the data, services, or resources of another computer or process on the network. Contrast with *server*.

client application thread. RPC: A thread executing client application code that makes one or more remote procedure calls. See *application thread*, *local*

application thread, *RPC thread*, and *server application thread*.

client context. RPC: The state within an RPC server generated by a set of remote procedures and maintained across a series of calls for a particular client. See *context handle*. See also *manager*.

client stub. RPC: The surrogate code for an RPC interface that is linked with and called by the client application code. In addition to general operations such as marshalling data, a client stub calls the RPC runtime to perform remote procedure calls and, optionally, to manage bindings. See *server stub*.

code page. (1) A table showing codes assigned to character sets. (2) An assignment of graphic characters and control function meanings to all code points. (3) Arrays of code points representing characters that establish numeric order of characters. [OSF] (4) A particular assignment of hexadecimal identifiers to graphic elements. (5) Synonymous with code set. (6) See also *code point*, *extended character*.

collapse. CDS: To remove the contents of a directory from the display (close it) using the CDS Browser. To collapse an open directory, double-click on its icon. Double-clicking on a closed directory expands it. Contrast with *expand*.

commit. To make all updates permanent. When an Encina transaction commits, all actions associated with that specific transaction have been written to the log. Even in the event of system problems, those actions are repeated if necessary when the system's recovery mechanism replays the log. See also *abort*.

context handle. RPC: A reference to state (client context) maintained across remote procedure calls by a server on behalf of a client. See *client context*.

control access. CDS: An access right that grants users the ability to change the access control on a name and to perform other powerful management tasks, such as replicate a directory or move a clearinghouse.

copy. GDS, XDS: Either a copy of an entry stored in other DSAs through bilateral agreement or a locally and dynamically stored copy of an entry resulting from a request (a cache copy).

coupling facility. A special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex.

credentials. Security: A general term for privilege attribute data that has been certified by a trusted privilege certification authority.

cross-linking information. In order for z/OS DCE to provide RACF-DCE interoperability and single sign-on

to DCE, DCE provides utilities (see **mvsexpt** and **mvsimpt**) to incorporate into RACF the information that associates a z/OS-RACF user ID with a DCE principal's identifying information and the DCE principal's UUID with the corresponding z/OS-RACF user ID. The information is placed in a RACF DCE segment and the RACF general resource class, DCEUUIDS. This is called **cross-linking information** and is what allows interoperability and single sign-on to work. See also *interoperability* and *single sign-on*.

cross-system coupling facility (XCF). A component of MVS that provides functions to support cooperation between authorized programs running within a sysplex.

Customer Information Control System (CICS). An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

D

daemon. (1) A long-lived process that runs unattended to perform continuous or periodic system-wide functions such as network control. Some daemons are triggered automatically to perform their task; others operate periodically. An example is the **cron** daemon, which periodically performs the tasks listed in the **crontab** file. Many standard dictionaries accept the spelling *demon*. (2) A DCE server process.

DCE. Distributed Computing Environment.

DFS. Distributed File Service.

directory. (1) A logical unit for storing entries under one name (the directory name) in a CDS namespace. Each physical instance of a directory is called a replica. (2) A collection of open systems that cooperates to hold a logical database of information about a set of objects in the real world.

directory ID. Directory identifier.

Directory Service. A DCE component. The Directory Service is a central repository for information about resources in a distributed system. See *Cell Directory Service* and *Global Directory Service*.

directory system. GDS: A system for managing a directory, consisting of one or more DSAs. Each DSA manages part of the DIB.

distributed computing. A type of computing that allows computers with different hardware and software to be combined on a network, to function as a single computer, and to share the task of processing application programs.

Distributed Computing Environment (DCE). A comprehensive, integrated set of services that supports the development, use, and maintenance of distributed applications. DCE is independent of the operating system and network; it provides interoperability and portability across heterogeneous platforms.

Distributed File Service (DFS). A DCE component. DFS joins the local file systems of several file server machines making the files equally available to all DFS client machines. DFS allows users to access and share files stored on a file server anywhere in the network, without having to consider the physical location of the file. Files are part of a single, global name space, so that a user can be found anywhere in the network by means of the same name.

distributed program link (DPL). CICS: An extension to the CICS API that allows a program to link to another program on a remote system.

Distributed Time Service (DTS). A DCE component. It provides a way to synchronize the times on different hosts in a distributed system.

DLL. Dynamic link library.

DPL. Distributed program link.

DTS. Distributed Time Service.

DTS entity. DTS: The server or clerk software on a system.

DUA cache. GDS: The part of the DUA that stores information to optimize name lookups. Each cache contains copies of recently accessed object entries as well as information about DSAs in the directory.

dynamic link library (DLL). Binds parts of the executable at load or runtime.

E

ECI. Externally callable interface.

EID. An AS IMS identifier consisting of the Encina GTID and TID associated with a given transaction.

element. RPC: Any of the bits of a bit string, the octets of an octet string, or the octets by means of which the characters of a character string are represented.

Encina. A family of software products for building and running large-scale, distributed client-server systems. Encina uses and enhances the facilities DCE provides with transactional semantics.

endpoint. RPC: An address of a specific server instance on a host.

endpoint map. RPC: A database local to a node where local RPC servers register binding information associated with their interface identifiers and object identifiers. The endpoint map is maintained by the endpoint map service of the DCE daemon.

endpoint map service. RPC: A service that maintains a system's endpoint map for local RPC servers. When an RPC client makes a remote procedure call using a partially bound binding handle, the endpoint map service looks up the endpoint of a compatible local server. See *endpoint map*.

entity. (1) CDS: Any manageable element through the CDS namespace. Manageable elements include directories, object entries, servers, replicas, and clerks. The CDS control program (CDSCP) commands are based on directives targeted for specific entities. (2) DTS: See *DTS entity*.

entry. GDS, XDS: The part of the DIB that contains information relating to a single directory object. Each entry consists of directory attributes.

ENV. environment variable

environment variable (ENV). A variable included in the current software environment that is available to any called program that requests it.

ephemeral application. An application that does not contain any recoverable data. See *recoverable data*.

exception. (1) An abnormal condition such as an I/O error encountered in processing a data set or a file. (2) One of five types of errors that can occur during a floating-point exception. These are valid operation, overflow, underflow, division by zero, and inexact results. [OSF] (3) Contrast with *interrupt, signal*.

executor thread. See *call thread*.

expand. CDS: To display the contents of (open) a directory using the CDS Browser. A directory that is closed can be expanded by double-clicking on its icon. Double-clicking on an expanded directory collapses it. Contrast with *collapse*.

export. (1) RPC: To place the server binding information associated with an RPC interface or a list of object UUIDs or both into an entry in a name service database. (2) To provide access information for an RPC interface. Contrast with *unexport*.

externally callable interface (ECI). CICS/ESA: A host version of the OS/2® CICS externally callable interface. It is used by the CICS adapter as an interface to CICS.

F

foreign cell. A cell other than the one to which the local machine belongs. A foreign cell and its binding information are stored in either GDS or the Domain Name System (DNS). The act of contacting a foreign cell is called intercell. Contrast with *local cell*.

full name. CDS: The complete specification of a CDS name, including all parent directories in the path from the cell root to the entry being named.

fully bound binding handle. RPC: A server binding handle that contains a complete server address including an endpoint. Contrast with *partially bound binding handle*.

G

GDS. Global Directory Service.

General-Use Programming Interface (GUIP). An interface, with few restrictions, for use in customer-written programs. The majority of programming interfaces are general-use programming interfaces, and are appropriate in a wide variety of application programs. A general-use programming interface requires the knowledge of the externals of the interface and perhaps the externals of related programming interfaces. Knowledge of the detailed design or implementation of the software product is not required.

Global Directory Service (GDS). A DCE component. A distributed replicated directory service that provides a global namespace that connects the local DCE cells into one worldwide hierarchy. DCE users can look up a name outside a local cell with GDS.

global name. A name that is universally meaningful and usable from anywhere in the DCE naming environment. The prefix /... indicates that a name is global.

group. (1) RPC: A name service entry that corresponds to one or more RPC servers that offer common RPC interfaces, RPC objects, or both. A group contains the names of the server entries, other groups, or both that are members of the group. See *NSI group attribute*. (2) Security: Data that associates a named set of principals that can be granted common access rights. See *subject identifier*.

GTID. Encina global transaction identifier (global TID). An identifier that is unique across all servers. This ties a transaction to all of its participating applications across different servers.

GUIP. General-Use Programming Interface.

H

handle. RPC: An opaque reference to information. See *binding handle*, *context handle*, *interface handle*, *name service handle*, and *thread handle*.

home cell. Synonym for *local cell*.

host ID. Synonym for *network address*.

I

identity mapping. Application Support server: A record in the Security Registry that contains the mapping between a client's DCE identity and an MVS user ID.

IDL. Interface Definition Language.

IDL compiler. RPC: A compiler that processes an RPC interface definition and an optional attribute configuration file (ACF) to generate client and server stubs, and header files. For Application Support, z/OS extends the IDL compiler to allow generation of server stubs for IMS or CICS transactions, COBOL data types, and Encina transactional RPCs. The z/OS IDL compiler option **-tidl** causes z/OS Encina Toolkit Executive TIDL preprocessing to occur transparently to the user, producing a transactional Application Support server stub.

IMS. Information Management System.

IMS control region. An IMS address space that controls the overall flow of information, execution, and data access within a set of IMS-dependent regions.

IMS recovery token. The identifier of a transaction for IMS.

Information Management System (IMS). A database and data communication system capable of managing complex databases and networks in virtual storage.

instance. XOM: An object in the category represented by a class.

interface. RPC: A shared boundary between two or more functional units, defined by functional characteristics, signal characteristics, or other characteristics, as appropriate. The concept includes the specification of the connection of two devices having different functions. See *RPC interface*.

interface definition. RPC: A description of an RPC interface written in the DCE Interface Definition Language (IDL). See *RPC interface*.

Interface Definition Language (IDL). A high-level declarative language that provides syntax for interface definitions.

For Application Support, z/OS extends IDL to support COBOL and C data types, Encina transactional attributes, and extensions to communicate with CICS and IMS.

interface handle. RPC: A reference in code to an interface specification. See *binding handle* and *interface specification*.

interface identifier. RPC: A string containing the interface Universal Unique Identifier (UUID) and major and minor version numbers of a given RPC interface. See *RPC interface*.

interface specification. RPC: An opaque data structure that is generated by the DCE IDL compiler from an interface definition. It contains identifying and descriptive information about an RPC interface. See *interface definition*, *interface handle*, and *RPC interface*.

interface UUID. RPC: The Universal Unique Identifier (UUID) generated for an RPC interface definition using the UUID generator. See *interface definition* and *RPC interface*.

Internet Protocol (IP). In TCP/IP, a protocol that routes data from its source to its destination in an Internet environment. IP provides the interface from the higher level host-to-host protocols to the local network protocols. Addressing at this level is usually from host to host.

interoperability. The capability to communicate, execute programs, or transfer data among various functional units in a way that requires the user to have little or no knowledge of the unique characteristics of those units.

interregion communication (IRC). CICS/ESA: The method by which CICS provides communication between a CICS region and another region in the same processor. Used for multiregion operation (MRO). See also *multiregion operation*.

intersystem communication (ISC). IMS/ESA: An extension of the IMS multiple systems coupling feature that permits the connection of IMS to another IMS subsystem, to CICS, or to a user-written subsystem, provided both systems use ISC.

IP. Internet Protocol

IRC. Interregion communication.

ISC. Intersystem communication.

K

Kerberos. The authentication protocol used to carry out DCE private key authentication. Kerberos was developed at the Massachusetts Institute of Technology.

key. A value used to encrypt and decrypt data.

L

local. (1) Pertaining to a device directly connected to a system without the use of a communication line. (2) Pertaining to devices that have a direct, physical connection. Contrast with *remote*.

local application thread. RPC: An application thread that runs within the confines of one address space on a local system and passes control exclusively among local code segments. See *application thread*, *client application thread*, *RPC thread* and *server application thread*.

local cell. The cell to which the local machine belongs. Synonymous with *home cell*. Contrast with *foreign cell*.

logical unit (LU). A host port through which a user gains access to the services of a network.

lookup context. The context setup by the client to locate compatible binding handles from the name space. Name service interfaces (NSI) are used to setup and free the lookup context.

LU. Logical unit.

M

manager. RPC: A set of remote procedures that implement the operations of an RPC interface and that can be dedicated to a given type of object. See also *object* and *RPC interface*.

mask. (1) A pattern of characters used to control the retention or deletion of portions of another pattern of characters (2) Security: Used to establish maximum permissions that can then be applied to individual ACL entries. (3) GDS: The administration screen interface menus.

message format service (MFS). An editing facility that allows application programs to deal with simple logical messages instead of device-dependent data, thus simplifying the application development process.

MRO. Multiregion operation.

multiregion operation (MRO). Communication between CICS regions in the same processor without

the use of SNA network facilities. It provides the user with the ability to share resources such as terminals, transactions, and data.

mutex. Mutual exclusion. A read/write lock that grants access to only a single thread at any one time. A mutex is often used to ensure that shared variables are always seen by other threads in a consistent way.

mvsexpt. One of two (the other is **mvsimpt**) utilities used to automate much of the administrator's work in creating the cross-linking information for DCE-RACF interoperability. The **mvsexpt** utility creates the cross-linking information in the RACF database from information in the DCE registry. See also *cross-linking information*, *interoperability*, and *single sign-on*.

mvsimpt. One of two (the other is **mvsexpt**) utilities used to automate much of the administrator's work in creating the cross-linking information for DCE-RACF interoperability. The **mvsimpt** utility creates DCE principals from information obtained from the RACF database. See also *cross-linking information*, *interoperability*, and *single sign-on*.

N

name. GDS, CDS: A construct that singles out a particular (directory) object from all other objects. A name must be unambiguous (denote only one object); however, it need not be unique (be the only name that unambiguously denotes the object).

name service handle. RPC: An opaque reference to the context used by the series of next operations called during a specific name service interface (NSI) search or inquiry.

namespace. CDS: A complete set of CDS names that one or more CDS servers look up, manage, and share. These names can include directories, object entries, and soft links.

network. A collection of data processing products connected by communications lines for exchanging information between stations.

network address. An address that identifies a specific host on a network. Synonymous with *host ID*.

Network Data Representation (NDR). RPC: The transfer syntax defined by the Network Computing Architecture. See *transfer syntax*.

network protocol. A communications protocol from the Network Layer of the Open Systems Interconnection (OSI) network architecture, such as the Internet Protocol (IP).

node. (1) An endpoint of a link, or a junction common to two or more links in a network. Nodes can be preprocessors, controllers, or workstations, and they can vary in routing and other functional capabilities. (2) In network topology, the point at an end of a branch. It is usually a physical machine.

NSI binding attribute. RPC: An RPC-defined attribute (NSI attribute) of a name service entry; the binding attribute stores binding information for one or more interface identifiers offered by an RPC server and identifies the entry as an RPC server entry. See *binding information* and *NSI object attribute*. See also *server entry*.

NSI group attribute. RPC: An RPC-defined attribute (NSI attribute) of a name service entry that stores the entry names of the members of an RPC group and identifies the entry as an RPC group. See *group*.

NSI object attribute. RPC: An RPC-defined attribute (NSI attribute) of a name service entry that stores the object UUIDs of a set of RPC objects. See *object*.

NSI profile attribute. RPC: An RPC-defined attribute (NSI attribute) of a name service entry that stores a collection of RPC profile elements and identifies the entry as an RPC profile. See *profile*.

NULL. In the C language, a pointer that does not point to a data object.

O

object. (1) A data structure that implements some feature and has an associated set of operations. (2) RPC: For RPC applications, anything that an RPC server defines and identifies to its clients using an object Universal Unique Identifier (UUID). An RPC object is often a physical computing resource such as a database, directory, device, or processor. Alternatively, an RPC object can be an abstraction that is meaningful to an application, such as a service or the location of a server. See *object UUID*. (3) XDS: Anything in the world of telecommunications and information processing that can be named and for which the directory information base (DIB) contains information. (4) XOM: Any of the complex information objects created, examined, changed, or destroyed by means of the interface.

object entry. CDS: The name of a resource (such as a node, disk, or application) and its associated attributes, as stored by CDS. CDS administrators, client application users, or the client applications themselves can give a resource an object name. CDS supplies some attribute information (such as a creation time stamp) to become part of the object, and the client application may supply more information for CDS to store as other attributes. See *entry*.

object identifier (OID). A value (distinguishable from all other such values) that is associated with an information object. It is formally defined in the CCITT X.208 standard.

object management (OM). The creation, examination, change, and deletion of potentially complex information objects.

object name. CDS: A name for a network resource.

object UUID. RPC: The Universal Unique Identifier (UUID) that identifies a particular RPC object. A server specifies a distinct object UUID for each of its RPC objects. To access a particular RPC object, a client uses the object UUID to find the server that offers the object. See *object*.

OID. Object identifier.

opaque. A datum or data type whose contents are not visible to the application routines that use it.

Open Software Foundation (OSF). A nonprofit research and development organization set up to encourage the development of solutions that allow computers from different vendors to work together in a true open-system computing environment.

Open Transaction Manager Access (OTMA). A transaction-based connectionless client-server protocol whose implementation is specific to IMS/ESA in an MVS sysplex environment. It uses the MVS Cross-System Coupling Facility (XCF) as its transport layer.

operation. (1) GDS: Processing performed within the directory to provide a service, such as a read operation. (2) RPC: The task performed by a routine or procedure that is requested by a remote procedure call.

organization. (1) The third field of a subject identifier. (2) Security: Data that associates a named set of users who can be granted common access rights that are usually associated with administrative policy.

OTMA. Open Transaction Manager Access.

P

PAC. Privilege attribute certificate.

partially bound binding handle. RPC: A server binding handle that contains an incomplete server address lacking an endpoint. Contrast with *fully bound binding handle*.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called

members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE). A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

password. A secret string of characters shared between a computer system and a user. The user must specify the character string to gain access to the system.

PCB. Program communication block.

PDS. Partitioned data set.

PDSE. Partitioned data set extended.

permission. (1) The modes of access to a protected object. The number and meaning of permissions with respect to an object are defined by the access control list (ACL) Manager of the object. (2) GDS: One of five groups that assigns modes of access to users: MODIFY PUBLIC, READ STANDARD, MODIFY STANDARD, READ SENSITIVE, or MODIFY SENSITIVE. Synonymous with *access right*. See also *access control list*.

pipe. (1) RPC: A mechanism for passing large amounts of data in a remote procedure call. (2) The data structure that represents this mechanism.

platform. The operating system environment in which a program runs.

PLT. Program list table.

position (within a string). XOM: The ordinal position of one element of a string relative to another.

position (within an attribute). XOM: The ordinal position of one value relative to another.

PPT. Processing program table.

presentation address. An unambiguous name that is used to identify a set of presentation service access points. Loosely, it is the network address of an open systems interconnection (OSI) service.

principal. Security: An entity that can communicate securely with another entity. In the DCE, principals are represented as entries in the Registry database and include users, servers, computers, and authentication surrogates.

privilege attribute. Security: An attribute of a principal that may be associated with a set of permissions. DCE

privilege attributes are identity-based and include the principal's name, group memberships, and local cell.

privilege attribute certificate (PAC). Security: Data describing a principal's privilege attributes that has been certified by an authority. In the DCE, the Privilege Service is the certifying authority; it seals the privilege attribute data in a ticket. The authorization protocol, DCE Authorization, determines the permissions granted to principals by comparing the privilege attributes in PACs with entries in an access control list.

privilege ticket. Security: A ticket that contains the same information as a simple ticket, and also includes a privilege attribute certificate. See *service ticket*, *simple ticket*, and *ticket-granting ticket*.

processing program table (PPT). CICS: A CICS table used to store program definition information, BMS mapset information, and partitionset information. Data stored includes programming language, program load characteristics, and initial execution availability. This table is used by CICS program control to locate, load, and maintain runtime statistics such as program size and usage.

profile. RPC: An entry in a name service database that contains a collection of elements from which name service interface (NSI) search operations construct search paths for the database. Each search path is composed of one or more elements that refer to name service entries corresponding to a given RPC interface and, optionally, to an object. See *NSI profile attribute* and *profile element*.

profile element. RPC: A record in an RPC profile that maps an RPC interface identifier to a profile member (a server entry, group, or profile in a name service database). See *profile*. See also *group*, *interface identifier* and *server entry*.

program communication block (PCB). IMS/ESA: A control block that describes the source or destination of messages, or the view of an IMS/ESA database.

program list table (PLT). CICS/ESA: A data area that contains a list of programs to be invoked.

programming interface. The supported method through which customer programs request software services. The programming interface consists of a set of callable services provided with the product.

protocol. A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication.

protocol sequence. Synonym for *RPC protocol sequence*.

Q

quiescent state. Application Support server: The server state wherein the server can process only management calls. The Application Support server enters this state when the server initialization has been completed or when the server is stopped by an administrative client.

R

RACF. Resource Access Control Facility.

RDO. Resource definition on-line.

read access. CDS: An access right that grants the ability to view data.

recoverable data. Data whose value persists across system shutdowns and failures. Changes made to recoverable data are permanent regardless of system problems. Logging changes to recoverable data is the most common method of ensuring permanence. Changes to data are recorded in a log that can be replayed to return the data to a valid state.

Recoverable Resource Management Services (RRMS). The z/OS system-level synchronization-point manager. Three system components provide the function: context services, registration services, and resource recovery services (RRS).

register. (1) RPC: To list an RPC interface with the RPC runtime. (2) To place server-addressing information into the local endpoint map. (3) To insert authorization and authentication information into binding information. See *endpoint map* and *RPC interface*.

remote. Pertaining to a device, file or system that is accessed by your system through a communications line. Contrast with *local*.

remote procedure. RPC: An application procedure located in a separate address space from calling code. See *remote procedure call*.

remote procedure call. RPC: A client request to a service provider located anywhere in the network.

Remote Procedure Call (RPC). A DCE component. It allows requests from a client program to access a procedure located anywhere in the network.

request. A command sent to a server over a connection.

resource. Items such as printers, plotters, data storage, or computer services. Each has a unique identifier associated with it for naming purposes.

Resource Access Control Facility (RACF). An IBM licensed program, that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, and logging the detected unauthorized access to protected resources.

Resource definition on-line (RDO). CICS/ESA: A series of on-line CICS system tables that tell the CICS system which resources to use, what their properties are, and how it can use them. CICS needs this information about system resources to run the system.

Resource Recovery Services (RRS). The system component that provides the services a resource manager calls to protect resources.

RPC. Remote Procedure Call.

RPC control program (RPCCP). An interactive administrative facility for managing name service entries and endpoint maps for RPC applications.

RPC interface. A logical group of operations, data types, and constant declarations that serves as a network contract for a client to request a procedure in a server. See also *interface definition* and *operation*.

RPC protocol. An RPC-specific communications protocol that supports the semantics of the DCE RPC API and runs over either connectionless or connection-oriented communications protocols.

RPC protocol sequence. A valid combination of communications protocols represented by a character string. Each RPC protocol sequence typically includes three protocols: a network protocol, a transport protocol, and an RPC protocol that works with the network and transport protocols. See *network protocol*, *RPC protocol*, and *transfer protocol*. Synonymous with *protocol sequence*.

RPC runtime. A set of operations that manages communications, provides access to the name service database, and performs other tasks, such as managing servers and accessing security information, for RPC applications. See *RPC runtime library*.

RPC runtime library. A group of routines of the RPC runtime that support the RPC applications on a system. The runtime library provides a public interface to application programmers, the application programming interface (API), and a private interface to stubs, the stub programming interface (SPI). See *RPC runtime*.

RPC thread. A logical thread within which a remote procedure call is executed. See *thread*.

RPCCP. RPC control program

RRMS. Recoverable Resource Management Services.

RRS. Resource Recovery Services.

S

SDSRM. Server distributed synchronization-point resource manager.

Security Service. A DCE component that provides trustworthy identification of users, secure communications, and controlled access to resources in a distributed system.

segment. One or more contiguous elements of a string.

server. (1) On a network, the computer that contains programs, data, or provides the facilities that other computers on the network can access. (2) The party that receives remote procedure calls. Contrast with *client*.

server application thread. RPC: A thread running the server application code that initializes the server and listens for incoming calls. See *application thread*, *client application thread*, *local application thread*, and *RPC thread*.

server entry. RPC: A name service entry that stores the binding information associated with the RPC interfaces of a particular RPC server and object Universal Unique Identifiers (UUIDs) for any objects offered by the server. See also *binding information*, *NSI binding attribute*, *NSI object attribute*, *object* and *RPC interface*.

server instance. RPC: A server running in a specific address space. See *server*.

server state. Application Support server: The condition of the Application Support server after it has been started. The server state may be any of the following, depending on the actions directed to it by the administrator: initializing, quiescent, starting, operating, or stopping.

server stub. RPC: The surrogate calling code for an RPC interface that is linked with server application code containing one or more sets of remote procedures (managers) that implement the interface. See *client stub*. See also *manager*.

service. In network architecture, the capabilities that the layers closer to the physical media provide to the layers closer to the end user.

service ticket. Security: A ticket for a specified service other than the ticket-granting service. See *privilege ticket*, *simple ticket*, and *ticket-granting ticket*.

session. GDS: A sequence of directory operations requested by a particular user of a particular directory user agent (DUA) using the same session object management (OM) object.

signal. Threads: To wake only one thread waiting on a condition variable. See *broadcast*.

sign-on. (1) A procedure to be followed at a terminal or workstation to establish a link to a computer. (2) To begin a session at a workstation. (3) Same as log on or log in.

simple name. CDS: One element in a CDS full name. Simple names are separated by slashes in the full name.

simple ticket. Security: A ticket that contains the principal's identity, a session key, a time stamp and other information, sealed using the target's secret key. See *privilege ticket*, *service ticket*, and *ticket-granting ticket*.

single sign-on. In z/OS DCE, single sign-on to DCE allows a z/OS user who has already been authenticated to an MVS external security manager, such as RACF, to be logged in to DCE. DCE does this automatically when a DCE application is started, if the user is not already logged in to DCE.

SIT. CICS/ESA: System initialization table.

specific. XOM: The attribute types that can appear in an instance of a given class, but not in an instance of its superclasses.

standard. A model that is established and widely used.

string. An ordered sequence of bits, octets, or characters, accompanied by the string's length.

stub. RPC: A code module specific to an RPC interface that is generated by the Interface Definition Language (IDL) compiler to support remote procedure calls for the interface. RPC stubs are linked with client and server applications and hide the intricacies of remote procedure calls from the application code. See *client stub* and *server stub*.

subject identifier (SID). A string that identifies a user or set of users. Each SID consists of three fields in the form person.group.organization. In an account, each field must have a specific value; in an access control list (ACL) entry, one or more fields may use a wildcard.

synchronization point. An intermediate or end point at which an update to one or more of the transaction's protected resources is logically complete and error-free during the processing of a transaction. A point in time from which IMS/ESA or CICS/ESA or an application

program can start over if a failure makes recovery necessary.

syntax. (1) XOM: An object management (OM) syntax is any of the various categories into which the OM specification statically groups values on the basis of their form. These categories are additional to the OM type of the value. (2) A category into which an attribute value is placed on the basis of its form. See *attribute syntax*.

sysplex. A set of MVS systems communicating and cooperating with each other to process customer workloads through certain multisystem hardware components and software services.

System initialization table (SIT). CICS/ESA: A CICS table that contains information to initialize and control system functions, module suffixes for selection of user-specified versions of CICS modules and tables, and information used to control the initialization process. A user can generate several SITs and select the one that best meets the user's requirements at initialization time.

system time. The time value maintained and used by the operating system.

T

TCP. Transmission Control Protocol

TCP/IP. Transmission Control Protocol/Internet Protocol

thread. A single sequential flow of control within a process.

thread handle. RPC: A data item that enables threads to share a storage management environment.

ticket. Security: An application-transparent mechanism that transmits the identity of an initiating principal to its target. See *privilege ticket*, *service ticket*, *simple ticket* and *ticket-granting ticket*.

ticket-granting ticket. Security: A ticket to the ticket-granting service. See *privilege ticket*, *service ticket*, and *simple ticket*.

TID. Transaction identifier. A unique name for a transaction. The Encina Tran-C component automatically manages passing, generating, and manipulating TIDs.

time provider (TP). DTS: A process that queries universal time coordinated (UTC) from a hardware device and provides it to the server.

TP. Time provider.

transaction. (1) A unit of processing consisting of one or more application programs initiated by a single request, often from a terminal. For Application Support, this refers to an IMS or a CICS transaction.

(2) IMS/ESA: A message destined for an application program.

transactional RPC. Similar to a standard DCE RPC but carrying additional information that identifies the Encina transaction on whose behalf it is running. A transactional RPC uses DCE RPCs as its underlying communication mechanism but extends this by providing transactional semantics.

transfer syntax. RPC: A set of encoding rules used for transmitting data over a network and for converting application data to and from different local data representations. See also *Network Data Representation*.

Transmission Control Protocol (TCP). A communications protocol used in Internet and any other network following the U.S. Department of Defense standards for inter-network protocol. TCP provides a reliable host-to-host protocol in packet-switched communication networks and in an interconnected system of such networks. It assumes that the Internet Protocol is the underlying protocol. The protocol that provides a reliable, full-duplex, connection-oriented service for applications.

Transmission Control Protocol/Internet Protocol (TCP/IP). A set of nonproprietary communications protocols that support peer-to-peer connectivity functions for both local and wide area networks.

two-phase commit protocol. A set of actions that ensures that an application program makes all the changes of a transaction to a collection of resources or makes no changes. The first phase is the prepare phase. One participant coordinates the transaction. After all the other participants have notified the coordinator that they have completed the prepare phase, the coordinator writes a prepare record. This ensures the transaction will complete, even if the system fails. The second phase is the commit phase. The coordinator notifies each participant to commit, and each writes a log record, releases any held resources, and sends the coordinator an acknowledgment. The coordinator informs all participants of the outcome and writes a commit record.

type. XOM: A category into which attribute values are placed on the basis of their purpose. See *attribute type*.

type UUID. RPC: The Universal Unique Identifier (UUID) that identifies a particular type of object and an associated manager. See also *manager* and *object*.

U

unexport. RPC: To remove binding information from a server entry in a name service database. Contrast with *export*.

Universal Unique Identifier (UUID). RPC: An identifier that is immutable and unique across time and space. A UUID can uniquely identify an entity such as an object or an RPC interface. See *interface UUID*, *object UUID*, and *type UUID*.

URID. Unit of Recovery ID. An IMS identifier of a transaction's RRS identity.

user. A person who requires the services of a computing system.

UUID. Universal unique identifier

V

value. XOM: An arbitrary and complex information item that can be viewed as a characteristic or property of an object. See *attribute value*.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VTAM. Virtual Telecommunications Access Method.

W

workstation. A device that enables users to transmit information to or receive information from a computer, for example, a display station or printer.

X

XDS. The X/Open Directory Services API.

X/Open Directory Services (XDS). An application program interface that DCE uses to access its directory service components. XDS provides facilities for adding, deleting, and looking up names and their attributes. The XDS library detects the format of the name to be looked up and directs the calls it receives to either GDS or CDS. XDS uses the XOM API to define and manage its information.

XCF. Cross-system coupling facility.

XOM. The X/Open OSI-Abstract-Data Manipulation API.

z/OS. A network computing-ready, integrated operating environment consisting of more than 50 base elements and integrated optional features delivered as a configured, tested system.

Bibliography

This bibliography is a list of publications for z/OS DCE and other products. The complete title, order number, and a brief description is given for each publication.

z/OS DCE Publications

This section lists and provides a brief description of each publication in the z/OS DCE library.

Overview

- *z/OS DCE Introduction*, GC24-5911

This book introduces z/OS DCE. Whether you are a system manager, technical planner, z/OS system programmer, or application programmer, it will help you understand DCE and evaluate the uses and benefits of including z/OS DCE as part of your information processing environment.

Planning

- *z/OS DCE Planning*, GC24-5913

This book helps you plan for the organization and installation of z/OS DCE. It discusses the benefits of distributed computing in general and describes how to develop plans for a distributed system in a z/OS environment.

Administration

- *z/OS DCE Configuring and Getting Started*, SC24-5910

This book helps system and network administrators configure z/OS DCE.

- *z/OS DCE Administration Guide*, SC24-5904

This book helps system and network administrators understand z/OS DCE and tells how to administer it from the batch, TSO, and shell environments.

- *z/OS DCE Command Reference*, SC24-5909

This book provides reference information for the commands that system and network administrators use to work with z/OS DCE.

- *z/OS DCE User's Guide*, SC24-5914

This book describes how to use z/OS DCE to work with your user account, use the directory service,

work with namespaces, and change access to objects that you own.

Application Development

- *z/OS DCE Application Development Guide: Introduction and Style*, SC24-5907

This book assists you in designing, writing, compiling, linking, and running distributed applications in z/OS DCE.

- *z/OS DCE Application Development Guide: Core Components*, SC24-5905

This book assists programmers in developing applications using application facilities, threads, remote procedure calls, distributed time service, and security service.

- *z/OS DCE Application Development Guide: Directory Services*, SC24-5906

This book describes the z/OS DCE directory service and assists programmers in developing applications for the cell directory service and the global directory service.

- *z/OS DCE Application Development Reference*, SC24-5908

This book explains the DCE Application Program Interfaces (APIs) that you can use to write distributed applications on z/OS DCE.

Reference

- *z/OS DCE Messages and Codes*, SC24-5912

This book provides detailed explanations and recovery actions for the messages, status codes, and exception codes issued by z/OS DCE.

z/OS SecureWay® Security Server Publications

This section lists and provides a brief description of books in the z/OS SecureWay Security Server library that may be needed for z/OS SecureWay Security Server DCE and for RACF® interoperability.

- *z/OS SecureWay Security Server DCE Overview*, GC24-5921

This book describes the z/OS SecureWay Security Server DCE and provides z/OS SecureWay Security Server DCE information about the z/OS DCE library.

- *z/OS SecureWay Security Server LDAP Client Programming*, SC24-5924

This book describes the Lightweight Directory Access Protocol (LDAP) client APIs that you can use to write distributed applications on z/OS DCE and gives you information on how to develop LDAP applications.

- *z/OS SecureWay Security Server RACF Security Administrator's Guide*, SA22-7683.

This book explains RACF concepts and describes how to plan for and implement RACF.

- *z/OS SecureWay Security Server LDAP Server Administration and Use*, SC24-5923

This book describes how to install, configure, and run the LDAP server. It is intended for administrators who will maintain the server and database.

- *z/OS SecureWay Security Server Firewall Technologies*, SC24-5922

This book provides the configuration, commands, messages, examples and problem determination for the z/OS Firewall Technologies. It is intended for network or system security administrators who install, administer and use the z/OS Firewall Technologies.

Tool Control Language Publication

- *Tcl and the Tk Toolkit*, John K. Osterhout, (c)1994, Addison—Wesley Publishing Company.

This non-IBM book on the Tool Control Language is useful for application developers, DCECP script writers, and end users.

IBM C/C++ Language Publication

- *z/OS C/C++ Programming Guide*, SC09-4765

This book describes how to develop applications in the C/C++ language in z/OS.

z/OS DCE Application Support Publications

This section lists and provides a brief description of each publication in the z/OS DCE Application Support library.

- *z/OS DCE Application Support Configuration and Administration Guide*, SC24-5903

This book helps system and network administrators understand and administer Application Support.

- *z/OS DCE Application Support Programming Guide*, SC24-5902

This book provides information on using Application Support to develop applications that can access CICS® and IMS™ transactions.

Encina Publications

- *z/OS Encina Toolkit Executive Guide and Reference*, SC24-5919

This book discusses writing Encina applications for z/OS.

- *z/OS Encina Transactional RPC Support for IMS*, SC24-5920

This book is to help software designers and programmers extend their IMS transaction applications to participate in a distributed, transactional client/server application.

Other Publications

This section lists and describes other books that you may need to use transactional RPCs.

- *IMS/ESA Open Transaction Manager Access Guide*, SC26-8743

This book describes OTMA clients, discusses IMS support for OTMA, and provides an overview of XCF for OTMA.

- *z/OS MVS Programming: Sysplex Services Guide*, SA22-7617

This book describes services that MVS provides for multisystem applications and subsystems.

- *z/OS MVS Programming: Sysplex Services Reference*, SA22-7618

This book provides syntax and parameter descriptions of MVS services for multisystem applications and subsystems.

- *IMS/ESA Operator's Reference*, SC26-8742

This book includes a list of commands supported from LU 6.2 (APPC) devices.

- *CICS Supplied Transactions*, SC34-5724

This book is for terminal operators (including master terminal operators), and for those who prepare procedures for terminal operators. It describes various transactions provided for terminal operators to use. These transactions perform tasks such as examining or changing CICS system status, obtaining statistics, and switching messages.

- *CICS System Programming Reference*, SC34-5726

This book provides reference material on the EXEC CICS INQUIRE/SET commands and other commands that form the CICS system programming interface. It includes information on examining and changing resource attributes, collecting resource statistics, and enabling and disabling an exit program.

- *CICS Recovery and Restart Guide*, SC34-5721

This book introduces the concepts of recovery and restart, and it describes the way that CICS recovery and restart mechanisms work. It gives guidance on determining your CICS recovery and restart needs, deciding which facilities are most appropriate, and implementing the chosen design on a CICS region. Topics covered include logging, journaling, user exits, user-written recovery procedures, communications error handling, programming considerations, data design for recovery, and the definition of the system.

- *z/OS MVS Programming: Resource Recovery*, SA22-7616

This book is for programmers who design and write authorized resource managers using the MVS synchronization-point manager, the z/OS Resource Recovery Services (RRS).

Index

A

abend codes, Application Support/CICS 273

Access Control List files

- database
 - creating 69
 - default naming convention 69
 - example 70
- default 68
- initial 68
- switching between current and backup 71
- version
 - creating 69

access, giving

- to namespace entries 64

accessing

- administration program 67
- CICS programs 37
- Identity Mapping files 60
- IMS programs 50

accounts

- DCE
 - creating for Application Support server 52
 - creating for user 55

ACID properties of Encina transactions 4

ACL Editor

- using to give access to administration program 72

ACL files

- database
 - creating 69
 - default naming convention 69
 - example 70
- default 68
- initial 68
- switching between current and backup 71
- version
 - creating 69

ACL Manager

- description 68

ACL, Initial Object 64

administering

- from console 115

administering Encina transactions 146

administration panels

- Application Support Server Administration Panel 101
- attaching Application Support Server with 102
- Create a CICS Application Support Server Definition Panel 109
- Create an IMS Application Support Server Definition Panel 110
- Display Interfaces and Server Operational Data Panel 103

administration panels (*continued*)

- flow 99
- Install / Uninstall Specific Interfaces Panel 104
- Perform DCE Login Panel 112
- Primary Options Panel 100, 102
- Work with Application Support Server Definitions Panel 108

administration program

- accessing 67, 72
- ACL database files
 - creating 69
 - example 70
 - file naming conventions for 69
 - switching between current and backup copies 71
 - version file, creating 69
- ACL permissions supported for 68
- ASUADMIN 74, 99
- configuring 99
- initial access control list 68
- starting 99

administrator

- creating DCE account for 55
- giving permissions for 67

administrator accounts

- creating DCE accounts for 55
- security subsystem 55

Application Support server

- access to, granting administrator 14, 67
- administering from the console 115
- administering, overview 13
- administration program
 - ACL permissions supported for 68
- administrator
 - creating DCE account for 55
- attaching to CICS or IMS subsystem 15, 16, 102, 115, 133
- attachment facility 8
- attachment to, stopping 103, 116, 134
- attributes, entering 76
- changing 16
- codes 195, 273
- common message variables 195
- creating CDS directories for 63
- creating DCE account for 51, 52
- creating envvar file for 21, 23
- creating home directory for 21
- defining to CDS
 - ASUADMIN administration program 99
 - steps 14, 74
- definitions 74, 108
- description 3
- giving access to RPCD 52

Application Support server *(continued)*

- home directory, creating 21
- illustration 3
- interface states 173
- interfaces, description 8
- keytab file for 51
- maintaining 17
- messages 195
- namespace entries, accessing 64
- operation, description 8
- restarting 16
- selecting group for 52
- server states 169
- specific message variables 195
- starting 14
- states
 - draining 173
 - initializing 169
 - installed 173
 - operational 170
 - quiescent 169
 - starting 170
 - stopping 170
 - uninstalled 173
- stopping 120

ASUADMIN 98

ASULUEE0 exit routine code listing 159

ASUVIFM 8, 107

ASUVPLOG 156

attaching

- Application Support server to CICS or IMS 115, 133

Attachment Facility for Application Support server 8

attributes

- CDS 167
- entering Application Support server 76

authorization groups

- creating 65

authorizing mechanisms

- CICS 38

B

BBF abend code 273

bibliography 293

books, list of DCE and related 293

C

CDS

- attributes of Application Support server 167
- defining Application Support server to 73, 74
- directories
 - creating for Application Support server 63
 - IMS and VTAM definition data in 50

CDS *(continued)*

- names 101
- namespace entries
 - configuring 14, 63
 - creating authorization groups for 65

Cell Directory Service

- attributes of Application Support server 167
- defining Application Support server to 73, 74
- directories
 - creating for Application Support server 63
 - IMS and VTAM definition data in 50
- names 101
- namespace entries
 - configuring 14, 63
 - creating authorization groups for 65

CICS

- Application Support security considerations 37
- applications
 - access control 37
- authorizing mechanisms in 38
- configuration worksheet example 77, 83
- configuring for Application Support server 33, 34
- defining transaction programs to 36
- installing Application Support server libraries 18
- programs
 - access to 37
- starting Application Support server 89
- with Application Support server 3

client

- allowed with Application Support server 3
- relationship between DCE client and Application Support server 3
- relationship between Encina client and Application Support server 5

code page, setting 32

codes, Application Support server 195

cold log start implications for Encina transactions 152

configuration worksheets

- blank 83—87
- example 77—81

configuring

- ASUADMIN administration program 99
- CDS namespace entries 14, 63
- CICS for Application Support server 13, 33
- IMS for Application Support server 13
- IMS with APPC for Application Support server 43
- IMS with ISC for Application Support server 47
- IMS with OTMA for Application Support Server 39
- MVS for Application Support server 18
- your system for the Application Support server, steps 13

CONNECTION definition for CICS 34

console, operator

- administering from 115

controlling
 message display 156

correlator ID, error 156

creating
 ACL database file 69
 authorization groups 65
 CDS directories for Application Support server 63
 DCE accounts
 for Application Support server 51, 52
 for Application Support server administrator 55
 for user 55
 requirements 51
 user accounts 13
 envar file 21, 23
 groups 52
 accessing Application Support server namespace
 entries 64
 home directory of Application Support server 21
 Identity Mapping file 57
 version file 69

D

daemons
 RPC 52

DCE
 accounts
 creating 55
 for Application Support server 13, 51, 52
 for Application Support server administrator 13
 requirements for creating 51
 user, creating 13
 AS CICS, startup procedure for 89
 AS IMS, startup procedure for 90, 91, 92, 93
 directory service 7
 environment variables, listing of 31
 security service 7
 user IDs
 for MVS 55
 mapping to MVS user IDs 56

default mapping 57

defining
 Application Support server to CDS 73, 74

definitions
 Application Support/IMS 47
 VTAM for Application Support/IMS 49

deleting
 Identity Mapping entries 60

DFHCSDUP utility
 using to generate CONNECTION definition 35
 using to generate SESSION definition 35

DFHMIRS 36

directory
 home, creating 21

displaying
 AS server's operational status 117, 128

displaying (*continued*)
 messages 156
 operational status of AS server 117, 128
 status of interfaces 119, 124, 126

draining state 173

E

Encina
 configuring for use with AS 37, 41
 description 4
 environment variables that AS uses 32
 evaluating 149
 services, how AS server uses 7
 transaction
 ACID properties 4
 administering 146
 description 4
 failure scenario example 149
 identifiers 141
 implications of restarting servers 152
 information, viewing 144
 intervening 147, 152
 locking 144
 phases and states 140
 transaction identifiers 141
 transaction processing, background 139
 transaction status 140

entering
 Application Support server attributes 76

envar file
 description 21, 23

environment variable file
 description 21, 23

environment variables
 _ASU_CICS_CONNECTIONS 27, 36
 _ASU_IDMAP 59
 _ASU_KEYTAB_FILE 51
 _EUV_HOME 22, 89, 90, 91, 92
 _EUV_RPC_ACL_FILE 71
 _EUV_SVC_MSG_LEVEL 155
 _EUV_SVC_MSG_LOGGING 156
 listing of 26

ephemeral client 7

ephemeral server 138

error correlator ID 156

error severity
 setting level of 155

example
 APPL definitions in VTAM 49
 AS CICS transaction 146
 ASUVSMDF IDL file 175
 configuration worksheet, blank 83—87
 configuration worksheet, sample 77—81
 Encina transaction failure scenario 149
 IMS TYPE and TERMINAL definition 47

example (continued)

- JCL for AS CICS 89
- JCL for AS IMS with APPC 91
- JCL for AS IMS with OTMA 90
- logical devices, defining 48
- startup PROC 22
- tkadmin abort transaction 147
- tkadmin force transaction 148
- tkadmin list transactions 144, 145, 146
- tkadmin query transaction 145, 146
- VTAM mode table entry 49
- VTAM, APPL definitions in 49

F

file

- ACL database 69
- envar, creating 21, 23
- example of ACL database 70
- Identity Mapping 56
- keytab 51
- version 69

flushing unsolicited IMS messages 156

G

giving access

- to namespace entries 64

glossary 279

granting access

- to namespace entries 64

groups

- authorization, creating 65
- creating for administrative purposes 52
- primary 52

groups, security 52

H

home directory

- creating and specifying 21

I

ID, error correlator 156

Identity Mapping file

- accessing 60
- changing entries in 60
- creating 57
- deleting entries in 60
- reloading 60, 113, 116, 132

IMS

- problem determination 156
- with APPC
 - configuration worksheet example 79, 85
 - configuring for Application Support server 43
 - definitions for Applications Support server 43
 - DFSLUEE0 configuration 46

IMS (continued)

with APPC (continued)

- starting Application Support server 91
- VTAM definitions for Applications Support server 44, 45

with Application Support server 3

with ISC

- configuration worksheet example 81, 87
- configuring for Application Support server 47
- definitions for Applications Support server 47
- flushing unsolicited messages 156
- security considerations 50
- starting Application Support server 92
- VTAM definitions for Applications Support server 49

with OTMA

- configuration worksheet example 80, 86
- configuring for Application Support server 39
- starting Application Support server 90
- with OTMA, configuring for Application Support Server 39

information, where to find more xiv

Initial Object ACL 64

initializing state 169

Installation

- Application Support server libraries 18

installation verification program (IVP) 14

installed state 173

installing interfaces 120, 130

interface

- ASUVIFM utility for installing 107
- description 8
- displaying status of 119, 124, 126
- for Application Support server 8
- installing 8, 103, 107, 120, 130
- prerequisites to installing 104
- uninstalling 8, 103, 107, 120, 130

interface states 173

IVP

- See installation verification program (IVP)

J

JCL

- sample, for AS CICS 89
- sample, for AS IMS with APPC 91
- sample, for AS IMS with OTMA 90

K

keytab file

- for Application Support server 51
- securing 51

L

- LC_ALL environment variable** 32
- libraries, installing Application Support server** 18
- locating**
 - error condition, client 156

M

- managing transactions called with transactional RPCs** 137
- mapping**
 - DCE principal names to MVS user IDs
 - with Identity Mapping file 56, 57
 - with RACF 60
 - default 57
 - specific 57
- Mapping file, Identity**
 - accessing 60
 - changing entries in 60
 - creating 57
 - deleting entries in 60
 - reloading 60, 113, 116, 132
- messages, Application Support server** 195
- migration** 11
- MODIFY operator command**
 - Identity Mapping entries 60
 - using to display server's operational status 117, 128
 - using to display status of interfaces 119
 - using to install interface 120
 - using to start attachment 115
 - using to stop Application Support server 120
 - using to stop attachment 115
 - using to uninstall interface 120
- MRO communication path, defining CICS** 34
- MVS**
 - CONNECTION definition for CICS 34
 - Cross System Coupling Facility (XCF) 13
 - installing Application Support server libraries 18
 - MRO communication path, defining CICS 34
 - naming conventions for CICS application IDs 35
 - security subsystem
 - registering users 55
 - SESSION attributes, CICS 35
 - user IDs, mapping DCE user accounts to 56

N

- namespace entries, CDS**
 - accessing 64
 - attributes of 167
 - configuring 63
 - giving access to 64
- naming conventions for CICS application IDs** 35

O

- operational state** 170
- operational status of AS server, displaying** 117, 128
- operator console**
 - administering from 115
- OTMA**
 - configuring 39
 - description 4
 - operational considerations 41

P

- partitioned data set** 119
- partitioned data set extended** 119
- passwords**
 - storing in keytab file 51
- PDS or PDSE** 119
- permissions**
 - ACL
 - supported for administration program 68
 - obtaining for RPCD 52
 - required by Application Support server administrator 64
- primary group** 52
- problem determination**
 - _EUV_SVC_MSG_LEVEL 155, 156
 - _EUV_SVC_MSG_LOGGING 156
 - using messaging facility 155
- program, administration**
 - accessing 67, 72
 - ACL database files
 - creating 69
 - example 70
 - file naming conventions for 69
 - switching between current and backup copies 71
 - version file, creating 69
 - ACL permissions supported for 68
 - ASUADMIN 74, 99
 - configuring 99
 - initial access control list 68
 - starting 99
- programs**
 - IMS
 - accessing 50

Q

- quiescent state** 169

R

- RACF** 56, 60
- reason codes** 273
- registering**
 - users with MVS security subsystem 55

Registry Editor 54
RELOAD 60, 132
RELOAD command 113
reloading Identity Mapping file 116, 132
requirements
 for creating a DCE account 51
 for nontransactional RPCs with OTMA 9
 for transactional RPCs 9
Resource Recovery Services (RRS) 5
RPC
 daemon
 giving Application Support server access to 52
 with Application Support server 7
RRS 5

S

sample JCL
 for AS CICS 89
 for AS IMS with APPC 91
 for AS IMS with OTMA 90
securing
 keytab file 51
security considerations
 Application Support/CICS 37
 Application Support/IMS 50
security groups 52
security service 7
security subsystem, MVS
 registering users 55
selecting
 group for Application Support server 52
server state reason codes
 Application Support/CICS 274
 Application Support/CICS and Application
 Support/IMS 274
 Application Support/IMS 275
server states 169
SESSION attributes for CICS 35
setting
 _ASU_IDMAP environment variable 59
 _ASU_KEYTAB_FILE environment variable 51
 _EUV_HOME environment variable 22
 _EUV_RPC_ACL_FILE environment variable 71
specific mapping 57
specifying
 home directory of Application Support server 21
starting
 Application Support server attachment 115, 133
starting state 170
state reason codes 273
states
 draining 173
 Encina transaction 140
 initializing 169
 installed 173

states (*continued*)
 operational 170
 quiescent 169
 starting 170
 stopping 170
 uninstalled 173
states, interface 173
states, server 169
stopping
 Application Support server 120
 Application Support server attachment 103, 116,
 134
stopping state 170

T

tkadmin abort transaction example 147
tkadmin command
 ACL determining access to 14
 displaying transactional information 17
 failure during network outage 147
 permission needed 68
 providing information on GTID and TID 142
 using 135
tkadmin force transaction example 148
tkadmin list transactions 142
tkadmin list transactions example 144, 145, 146
tkadmin query transaction 142
tkadmin query transaction example 145, 146
transaction programs, defining to CICS 36
transactional RPCs
 description 4
 managing transactions called with 137
two-phase commit protocol 4, 139

U

uninstalled state 173
uninstalling interfaces 120, 130
user
 creating DCE account for 55
 registering with MVS security subsystem 55
user ID
 mapping DCE to MVS 56

V

variables
 common to all messages 195
 specific to some messages 195
variables, environment
 _ASU_CICS_CONNECTIONS 27, 36
 _ASU_IDMAP 59
 _ASU_KEYTAB_FILE 51
 _EUV_HOME 22, 89, 90, 91, 92
 _EUV_RPC_ACL_FILE 71
 _EUV_SVC_MSG_LEVEL 155

variables, environment *(continued)*
 _EUV_SVC_MSG_LOGGING 156
 listing of 26
variant characters 32
vensrv_mgmt_display_if 124
vensrv_mgmt_display_ifs 126
vensrv_mgmt_display_server 128
vensrv_mgmt_if_mgmt 8, 130
vensrv_mgmt_reload_int 132
vensrv_mgmt_start_attachment 133
vensrv_mgmt_stop_attachment 134
VTAM
 definitions 49

W

worksheets
 configuration, blank 83—87
 configuration, example 77—81

X

XCF 13

Z

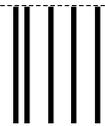
z/OS Encina Toolkit Executive 4



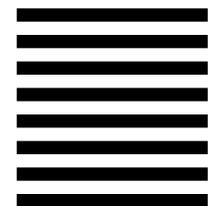
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Department G60
International Business Machines Corporation
Information Development
1701 North Street
ENDICOTT NY 13760-5553



Fold and Tape

Please do not staple

Fold and Tape

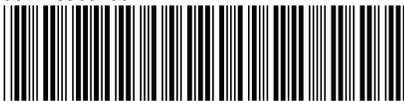


Program Number: 5694-A01



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC24-5903-00





z/OS DCE Application Support

Configuration and Administration Guide