iWay

iWay Server Configuration and Operations for MVS
Version 5 Release 2.0

DN3501029.0103

EDA, EDA/SQL, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, FOCUS Vision, Hospital-Trac, Information Builders, the Information Builders logo, Parlay, PC/FOCUS, SmartMart, SmartMode, SNAPpack, TableTalk, WALDO, Web390, WebFOCUS and WorldMART are registered trademarks, and iWay and iWay Software are trademarks of Information Builders, Inc.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2003, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

Preface

This manual describes the steps for configuring the iWay Server for MVS Version 5 Release 2.0. This manual also describes how to control the iWay Server for MVS with both the IBI Subsystem and the Console. This manual is intended for the person(s) who will be configuring and operating the server. It assumes that those persons have a working knowledge of operating systems and database management and have installed and configured the iWay Server for MVS.

How This Manual Is Organized

This manual includes the following chapters:

Chapter		Contents
1	*Configuring an MVS Server*	Describes the configuration process and the issues and options you may encounter during this process.
2	*Preparing for Server Configuration*	Explains the system resources (APPLIDS for VTAM or port numbers for TCP/IP) your system administrator must define and includes a Configuration Worksheet.
3	*Configuring Data Adapters*	Describes the steps required to configure various data adapters.
4	*Generating a Server*	Provides editing instructions for server configuration file jobs, and a description of how the resulting configuration files interact.
5	*Configuring Server System Features*	Explains how to configure the server system features that allow you to optimize and customize your server environment.
6	*Testing Server Communications*	Describes how to modify the startup JCL and submit it to the server.
7	*Configuring Supported Services*	Explains the steps required to configure server supported services.
8	*Modifying the Server Configuration*	Describes how to modify the parameters generated by the EDACFGX procedure in the EDASERVE file in qualif and EDACSG. Provides reference information on the structure of iWay's communications configuration files.

Chapter		Contents
9	*Accessing the Server*	Describes how to start and stop the server.
10	*IBI Subsystem*	Describes how to utilize the IBI Subsystem.
11	*Using the Server Console*	Describes how to manage server operations using the Server Console.

Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
`THIS TYPEFACE` or `this typeface`	Denotes syntax that you must enter exactly as shown.
`this typeface`	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
`underscore`	Indicates a default setting.
this typeface	Represents a placeholder (or variable) in a text paragraph, a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option you can click or select.
this typeface	Highlights a file name or command in a text paragraph that must be lowercase.
Key + Key	Indicates keys that you must press simultaneously.
`{ }`	Indicates two or three choices; type one of them, not the braces.
`[ ]`	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
`|`	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
`...`	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (…).

Convention	Description
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

See the Technical Publications Catalog for the most up-to-date listing and prices of technical publications, plus ordering information. To obtain a catalog, contact the Publications Order Department at (800) 969?4636.

You can also visit our World Wide Web site, http://www.iwaysoftware.com, to view a current listing of our publications and to place an order.

Customer Support

Do you have questions about iWay Server Configuration and Operations for MVS?

Call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your iWay Server Configuration and Operations for MVS questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code (*xxxx.xx*) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, http://www.informationbuilders.com. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions most effectively, please provide the following information when you call:

- Your six-digit site code (*xxxx.xx*).
- The exact nature of the problem:
  - Are the results or the format incorrect; are the text or calculations missing or misplaced?
  - The error message and return code, if applicable.
  - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two databases, have you tried executing a query containing just the code to access the database?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Documentation Services staff welcomes your opinions regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Documentation Feedback form on our Web site, http://www.iwaysoftware.com.

Thank you, in advance, for your comments.

iWay Software Training and Professional Services

Interested in training? Our Education Department offers a wide variety of training courses for iWay Software and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (http://www.iwaysoftware.com) or call (800) 969-INFO to speak to an Education Representative.

Interested in technical assistance for your implementation? Our Professional Services department provides expert design, systems architecture, implementation, and project management services for all your business integration projects. For information, visit our World Wide Web site (http://www.iwaysoftware.com).

Contents

1. Configuring an MVS Server 1-1
Hub Server 1-2
Full-Function Server 1-3
Full-Function Server (Not Using Hub Services) 1-3
Full-Function Server (Using Hub Services) 1-4
Communications Gateway 1-5

2. Preparing for Server Configuration 2-1
Product Certification 2-2
Hardware/Software Requirements for Server Configuration 2-2
Server Upgrade Considerations 2-5
Server Configuration Worksheets 2-6
Hub Server Configuration Worksheet 2-6
Full-Function Server Configuration Worksheet 2-10
Creating Network Definitions 2-15

3. Configuring Data Adapters 3-1
Data Adapter Configuration Considerations 3-2
Generating the Data Adapter for ADABAS 3-2
Generating the Data Adapter for CA-DATACOM/DB 3-3
Generating the Data Adapter for CA-IDMS/DB 3-7
Link-Editing User-Written Programs for the Data Adapter for IDMS 3-8
Implementing the ZBIND Exit From the Data Adapter for IDMS 3-8
Implementing the ZREADY Exit From the Data Adapter for IDMS 3-12
Generating the Data Adapter for DB2 3-14
Specifying an NCRS and NSTMTS Number Greater Than 99 3-16
Granting Access to DB2 3-16
Customizing the DB2 Security Exit 3-17
Generating the Data Adapter for IDMS/SQL 3-21
Customizing the Server JCL for IDMS/SQL 3-22
Allocating Data Sets for CV Access to IDMS/SQL 3-23
Allocating IDMS/SQL Data Sources for Local Mode Access 3-23

Customizing the IMS/BMP Environment 3-23
Accessing the XMI Server 3-24
Allocating the FOCBMP Communications File 3-25
Initiating the XMI Server in BMP Mode 3-26
Initiating the XMI Server in DLI Mode 3-27
Initiating the XMI Server With Data Sources Under Control of CICS 3-29
Invoking the Server in the XMI Server Environment 3-30
Allocating the Common Communications File for the XMI Server 3-31
Creating a PSB for the XMI Server 3-32
Terminating an XMI Server 3-32
Customizing the IMS/DBCTL Environment 3-34
Establishing Security 3-40
Configuring the Data Adapter for InfoMan 3-41
PICA Parameters for the Data Adapter for InfoMan 3-44
IGNORE Parameters in the InfoMan Access File 3-46
Generating the Data Adapter for Millennium 3-47
Software Requirements for the Data Adapter for Millennium 3-47
Customizing the Data Adapter for Millennium 3-47
Installing the Data Adapter for Millennium 3-51
Creating the Access File Library 3-51
Linking the Millennium API Modules 3-52
Generating the Data Adapter for MODEL 204 3-54
Generating the Data Adapter for Nomad 3-55
Generating the Data Adapter for Oracle 3-57
Generating the Data Adapter for SUPRA 3-58
Configuring the SUPRA Multi-Session Option 3-59
Configuring the SUPRA Multi-Session Batch Autostart Facility 3-62
Generating the Data Adapter for SYSTEM 2000 3-64
Generating the Data Adapter for Teradata 3-65

4. Generating a Server 4-1
Generating a Server Type 4-2
Specifying Values for an Ownerid 4-3
Generating a Hub Server 4-4
DDNAME LUPOOL 4-12
Generating a Full-Function Server 4-12
Next Steps 4-22

5. Configuring Server System Features 5-1
IBI Subsystem-Based Products 5-2
Benefits of Using the IBI Subsystem 5-2
How the IBI Subsystem Works 5-3
Installing the IBI Subsystem 5-3

Installing and Configuring HiperEDA 5-6
Hardware and Software Requirements for HiperEDA 5-6
HiperEDA Configuration 5-6
HiperBUDGET and the IBI Subsystem 5-8
Configuring the MVS Server for National Language Support 5-9
Changing the MVS Server Code Page Settings 5-11
Configuring MVS Server NLS Default Characteristics 5-16
Configuring Customized MVS Server NLS Monocasing 5-18
Configuring for Customized MVS Server NLS Sort Sequences 5-20
Modifying the Server JCL 5-22
Customizing MVS Client NLS Services 5-22
Configuring a Communications Gateway 5-24
Configuring the Transaction Server for IMS 5-27
CALLIMS Processing 5-28
Overview: Transaction Server for IMS Configuration 5-29
Step 1: Ensure That You Have Required Hardware and Software 5-29
Step 2: Determine the APPC/MVS Setup 5-29
Step 3: Set the Security Level 5-30
Step 4: Link-edit the API 5-30
Step 5: Confirm Interaction With IMS/TM 5-31
Step 6: Verify Installation of IMS/TM 5-31

6. Testing Server Communications 6-1
Modifying the Start-up JCL 6-2
Loading Sample Files 6-4
Running the Installation Verification Program (IVP) 6-4
Next Steps 6-7

7. Configuring Supported Services 7-1
FOCUS Database Server Support 7-2
Resource Analyzer/Resource Governor Software Components 7-5
Configuring Resource Analyzer and Resource Governor With Your Server 7-6
Granting Access to Resource Analyzer/Resource Governor Databases 7-13
Running the Resource Governor Configuration Verification Program 7-14
Using Resource Analyzer/Resource Governor Trace Files 7-16
Enabling Auto Procedures for an MVS Server 7-16
Next Steps 7-19

8. Modifying the Server Configuration 8-1
EDACFG Configuration Files 8-2
Changing Your Server Configuration 8-3
Setting Server Security 8-3
Using SAF and External Security Packages 8-4
Server Security Interfaces 8-5
How Server Security Works 8-5
Enabling Usage Accounting 8-8
Configuring the Server Console 8-8
Setting Server Console Security 8-10
Setting Internal Security for the Server Console 8-10
Setting External Security for the Server Console 8-12
Configuring IMS/DBCTL Access 8-15
Configuring Services for Inbound Protocols 8-15
Modifying the Communications Subsystem Configuration File 8-20
Protocol and Node Blocks in a Communications Subsystem Configuration File 8-22
Configuring Inbound Communications 8-25
Configuring Outbound Communications 8-30

9. Accessing the Server 9-1
Starting the Server 9-2
Stopping the Server 9-2

10. IBI Subsystem 10-1
Operating the IBI Subsystem With the SUBSYSI Program 10-2
Interacting With the IBI Subsystem 10-3
Displaying IBI Subsystem Information 10-4
Controlling the Hiperspace Usage of the IBI Subsystem 10-4
SET Commands 10-4
Troubleshooting 10-6

11. Using the Server Console 11-1
Accessing the Server Console 11-2
Selecting Server Console Commands 11-4
Searching for Allocations 11-21
Executing MVS Operator Commands 11-23
Terminating Connector Connections 11-25
Changing the Priority of the Subtask 11-27
Controlling the DBCTL Environment 11-28
Setting Dump Support Characteristics 11-29
Enabling Connector Connections 11-31
Server Monitoring and Statistics 11-32
Preventing New Connections 11-36
MVS Job Tracing 11-37

CHAPTER 1

Configuring an MVS Server

Topics:

- Hub Server
- Full-Function Server
- Communications Gateway

The configuration process builds the configuration files that allow a client application to connect to your server. It establishes connectivity between the application and data that resides on a server or subserver. When you have finished the configuration process, an application is able to connect to a data source.

During the configuration process, you define the server type and set up communications between the components. The server type dictates how a server accesses data. Setting up communications consists of defining the communications protocols on which the server receives and transmits requests and data to various other services.

Server types are:

- Hub Server.
- Full-Function Server, with or without Hub Services.
- Communications Gateway.
- ETL Server, with or without Hub Services.

Your choice of a server type determines the process through which your server accesses data. Each server type has a different data access process and related configuration file considerations. By selecting the appropriate server type for each of the servers in your network, you acquire access to all your data with minimal overhead. For details, see *Hub Server* on page 1-2, *Full-Function Server* on page 1-3, and *Communications Gateway* on page 1-5.

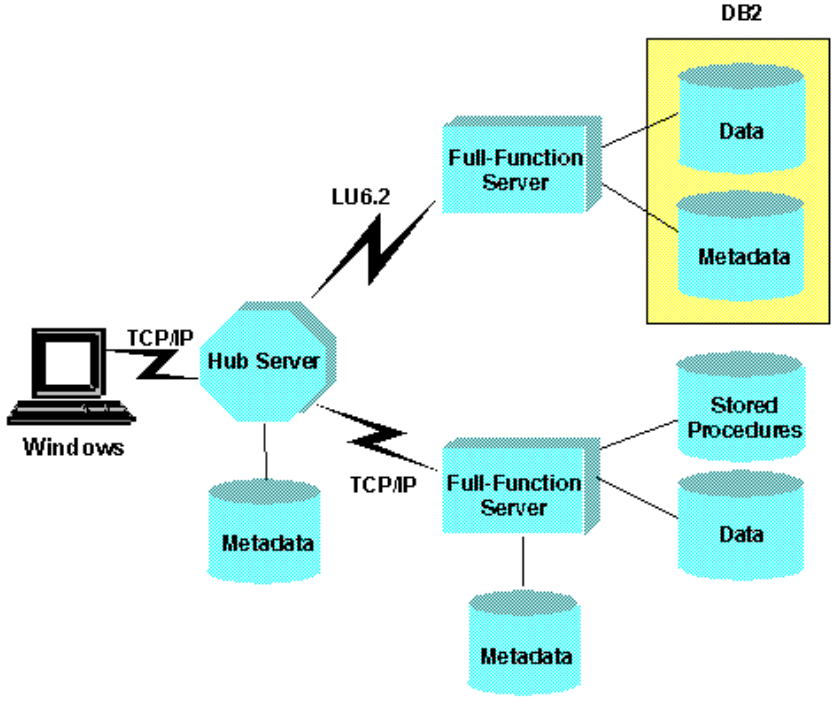
Hub Server

A Hub Server acts as a central point for distributing requests and gathering answer sets. It allows a client application using a single connection to receive data from multiple data sources across servers. A Hub Server allows you to:

- Connect servers across platforms.
- Perform cross-server join operations between both relational and non-relational data.
- Have SQL dialect translation.
- Use location transparent routing.
- Enable single-point security authentication.
- Use protocol translation, when necessary.

By performing these tasks, the Hub Server provides the application with location transparency of data.

The following figure illustrates a distributed processing configuration in which a Windows client is connected to a Hub Server, which, in turn, is connected to a Relational Gateway and a Full-Function Server.



In this configuration, the Hub Server is distributing requests for remote data and stored procedures. There is no local data access. The client sends SQL requests to the Hub Server using TCP/IP. The Hub Server receives these requests and routes them outbound using LU6.2 or TCP/IP to the available subserver. To configure a Hub Server, proceed to Chapter 2, *Preparing for Server Configuration*.

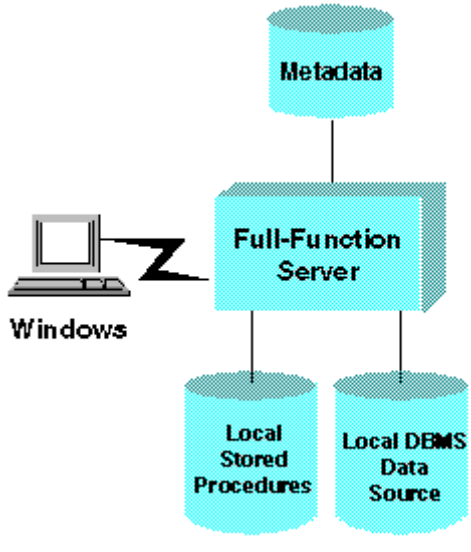
Full-Function Server

A Full-Function Server provides access to any supported relational or non-relational data source. It performs joins between relational and non-relational data sources. It also supports the use of Remote Procedure Calls.

You can use Hub Services to access data across platforms and provide location transparency to end users. For more details, see *Full-Function Server (Using Hub Services)* on page 1-4.

Full-Function Server (Not Using Hub Services)

You configure a Full-Function Server without Hub Services if the relational and non-relational data sources you want to access reside locally. The following figure illustrates a Full-Function Server accessing relational and non-relational data sources and stored procedures. Hub Services is not used in this example.

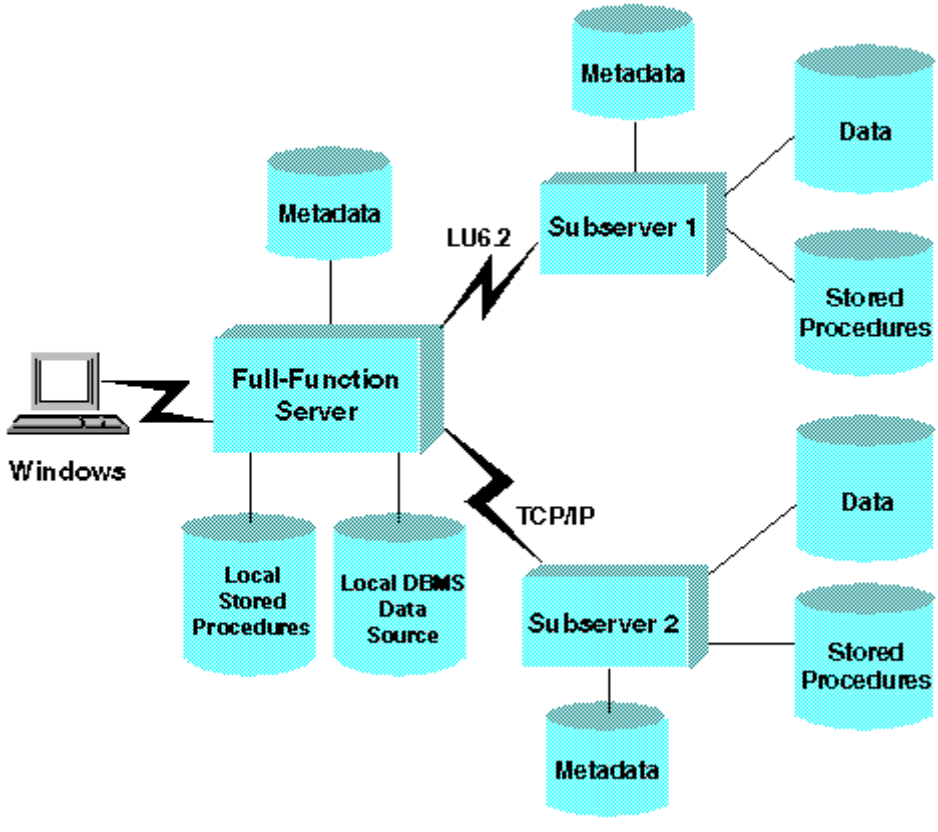


The Full-Function Server retrieves the data from the data sources, performs joins if necessary, and returns the answer set to the client. To configure a Full-Function Server, proceed to Chapter 2, *Preparing for Server Configuration*.

Full-Function Server (Using Hub Services)

A Full-Function Server that uses Hub Services assumes the functionality of a Hub Server in addition to the functionality of a Full-Function Server. It can access relational and non-relational data, both locally and remotely, while providing location transparency for the user.

The following figure illustrates a Full-Function Server acting as both a Hub Server and a server of local data and stored procedures. Clients can access data on the Full-Function Server or either of the subservers, or join data across any combination of servers or subservers.



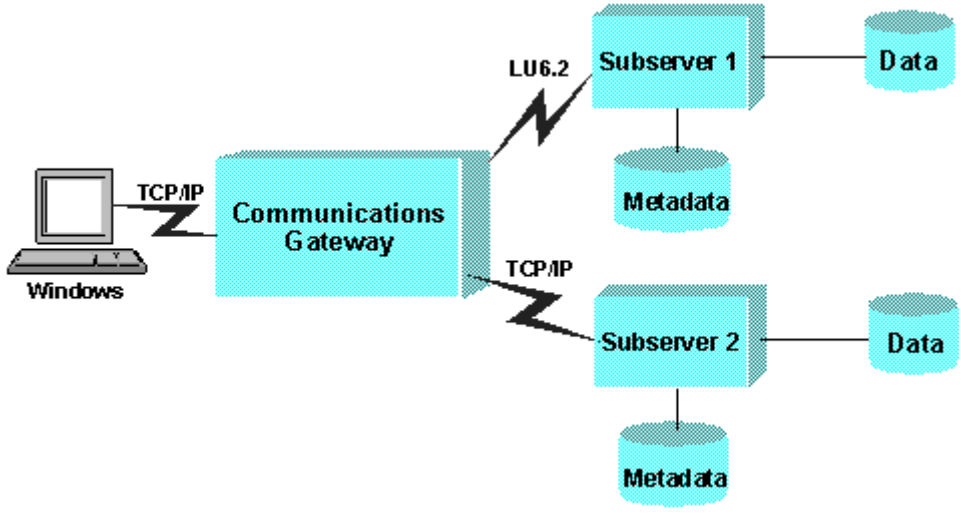
In this configuration, the Full-Function Server uses Hub Services to distribute requests for remote data and stored procedures to subservers. These subservers can reside on any supported platform and use any supported communications protocol. Additionally, the Full-Function Server can handle requests for local data and stored procedures. To configure a Full-Function Server, proceed to Chapter 2, *Preparing for Server Configuration*.

Communications Gateway

You can configure a server as a Communications Gateway. A Communications Gateway performs protocol translation only.

Note: On the server, the Communications Gateway is called EDAGATE. In addition to being an independent server, you also have the option of configuring it as part of a Hub or Full-Function Server (with Hub Services).

The following figure illustrates a server acting as a Communications Gateway. The server receives an SQL request from a client application using TCP/IP. The server, configured as a Communications Gateway, translates the protocol from TCP/IP to LU6.2. It then passes the request to the subserver where the DBMS resides for processing. The DBMS processes the request and then returns it along the same route. Note that a Communications Gateway does not perform any SQL processing. It simply distributes requests to the appropriate server. To configure a Communications Gateway, proceed to Chapter 2, *Preparing for Server Configuration*.



CHAPTER 2

Preparing for Server Configuration

Topics:

- Product Certification
- Hardware/Software Requirements for Server Configuration
- Server Upgrade Considerations
- Server Configuration Worksheets
- Creating Network Definitions

Before you begin to configure servers for MVS, be sure to review the following information:

- Products that are certified with the current production release of the server. See *Product Certification* on page 2-2.
- Hardware and software required for configuring a server. See *Hardware/Software Requirements for Server Configuration* on page 2-2.
- Supported communications protocols, including protocol-specific hardware and software requirements. See *Communications Protocols* on page 2-3.
- Upgrade considerations. See *Server Upgrade Considerations* on page 2-5.
- System resources required for network definitions. See *Creating Network Definitions* on page 2-15.

In addition, in preparation for configuration, we urge you to complete a series of Configuration Worksheets that are designed to help you during the configuration process. See *Server Configuration Worksheets* on page 2-6.

Product Certification

The following components are certified with full production status for the MVS Full-Function Server:

- Resource Analyzer/Resource Governor.
- Transaction Server for IMS.
- Hub Server.
- Full-Function Server supporting:

ADABAS	CA-Datacom	CA-IDMS	CA-IDMS/SQL
FOCUS Services	DB2	FOCUS	FOCSAM (VSAM, QSAM)
Model 204	IMS	Information Manager/2 (Infoman)	Millennium
SUPRA	Nomad	ORACLE	System 2000
Teradata	TOTAL		

Hardware/Software Requirements for Server Configuration

In the MVS environments, an IBM® or IBM-compatible mainframe that supports the following software is required to configure a server:

- JES2 Version 3.x or higher.
- MVS/SP V3 or higher.
- OS/390 1.x or higher.
- z/OS 1.0 or higher.
- A data adapter, as required by your site.

Additional hardware and software is required for specific communications protocols. See *Communications Protocols* on page 2-3.

Reference Communications Protocols

In the MVS environments, the following protocols are supported:

Protocol	Description
LU6.2	For users of SNA LU6.2 or APPC.
TCP/IP Sockets	For users of any communications capability on the server that is addressable using standard sockets calls (for example, IBM TCP/IP, Interlink, and OCS TCP/IP).
TCP/IP (Network Connect Systems)	For users of TCP/IP (Network Connect Systems).

The mainframe hardware and software required to configure a server for a specific communications protocol are:

Protocol	Hardware	Software
LU6.2	An SNA network and network devices capable of providing LU6.2 services to a host.	ACF/VTAM Version 3.2, PUT Level 9001 or higher. Server Communications Protocol Interface for LU6.2.
IBM TCP/IP	An IBM 3172 Interconnect Controller, or an IBM 8232 LAN Channel Station to connect an Ethernet network to the IBM mainframe.	IBM TCP/IP for MVS, Version 3.1 or higher. Server Communications Protocol Interface for TCP/IP.

Protocol	Hardware	Software
Interlink TCP/IP	You must have one of the following: • ACC ACS 9310 (Ethernet/802.3). • ACC IF-370/DDN (X.25 or HDH), ACC's 9315 (Ethernet/802.3). • BTI ELC ELC2 running 8232 emulation microcode. • IBM 3172 Interconnect Controller (Ethernet/802.3). • IBM 8232 LAN Channel Station (Ethernet/802.3). • Interlink 3722 (Ethernet/802.3). • Interlink 3732 (Ethernet/802.3). • Interlink 3752 (Ethernet/802.3). • Intel 87xx Fastpath Network Interfaces (Ethernet/802.3). • McData Linkmaster 6100E (Ethernet/802.3). • Network Systems Corporation A222, A223, and N220 (HYPERchannel) adapters.	Either IBM ACF/VTAM Version 2 or higher, or Interlink SNS/TCP Access TCP/IP for MVS, Version 2.1 or higher. Server Communications Protocol Interface for TCP/IP.

Note: If you are using Interlink TCP/IP, you must apply the following Program Temporary Fixes (PTF) from Interlink:

Interlink Version	PTF
2.1	TP04206
3.1	TP04210
4.1	TP04208
5.2	No known PTFs required.

Server Upgrade Considerations

If you are upgrading from a Version 4.2.x, 4.3.x, or 5.1.x server and you will not be using any new features provided in Version 5.2, perform the steps required for the type of server you are configuring.

If you intend to use new features provided by the Version 5.2 server, you must complete the full installation process.

Procedure **How to Upgrade a Hub Server**

To configure a Hub Server:

1. Unload the tape/cartridge.
2. Run EDAJINS2.
3. Use/copy old EDACSG (changing protocol resources if necessary).
4. Use/copy old EDASERVE and, for 4.2.x and 4.3.x servers only, add keyword LICENCE= with your new three-part license key provided in the server package. This new keyword goes in the global block, before any service blocks.
5. Use/copy old EDASPROF/EDAPROF profiles.
6. Use/copy old server JCL (changing library names where appropriate).

Procedure How to Upgrade a Full-Function Server

To configure a Full-Function Server:

1. Unload the tape/cartridge.
2. Run EDAJINS2.
3. Use/copy old EDACSG (changing protocol resources if necessary).
4. Use/copy old EDASERVE and, for 4.2.x and 4.3.x servers only, add keyword LICENCE= with your new three-part license key provided in the server package. This new keyword goes in the global block, before any service blocks.
5. Use/copy old EDASPROF/EDAPROF profiles.
6. Generate new data adapters as appropriate.
7. Use/copy old server JCL (changing library names where appropriate).

Server Configuration Worksheets

The Configuration Worksheets are included in your documentation to help you gather, in advance, information that will be required during the configuration process. Each worksheet contains a list of parameters for which you need to provide values during configuration. It also contains a parameter description and default, if one exists.

Each server type has a corresponding set of Configuration Worksheets. We recommend that you fill out the appropriate worksheets for each instance of the server.

Prior to completing the server configuration worksheets, you must obtain information related to system resources, such as APPLIDs or port numbers, from a system administrator. For details, see *Creating Network Definitions* on page 2-15.

Hub Server Configuration Worksheet

If you are configuring a Hub Server:

- Complete all relevant sections of the Hub Server Configuration Worksheet.
- After you complete the worksheet, proceed to Chapter 4, *Generating a Server*. Use the values in your worksheets to edit *qualif*.EDALIB.DATA(EDACFGH). This member creates the configuration files for the Hub Server.
- If you use FOCUS data sources, you must install the IBI subsystem. See Chapter 5, *Configuring Server System Features*, for more information.

Reference Hub: Configuration Parameters Worksheet

Parameter	Description	Default Value	Supply Your Value Here
qualif	The high-level qualifier for the data sets.		
OWNER	The Owner ID for the Dynamic Catalog views and tables. It is usually the TSO user ID of the person configuring the server.		
DUNIT	The disk unit used when allocating the data sets.		
SERVTYPE	The server type. Do not change this value.	HUB	HUB
LICENSEK	The license number supplied with this product on the insert.		
MAXIMUM	The maximum number of users for this Hub Server. It should not exceed the license agreement.		
DEPLOY	The deployment type for this server. Values can be PRIVATE, ISOLATED, or POOLED.	PRIVATE	
SYSOUT	The SYSOUT class the server uses for trace files.		

Reference Hub: Console Parameters Worksheet (Optional)

Parameter	Description	Default Value	Supply Your Value Here
CONAPPLD	The VTAM resource name (APPLID) used for Console access.		

Reference Hub: Inbound TCP/IP Parameters Worksheet

Parameter	Description	Default Value	Supply Your Value Here
TCPNAME	**For IBM TCP/IP:** The name of the started task or the JOB name for the TCP/IP subsystem address space, if it is different from the default (TCPIP). **For Interlink TCP/IP:** The name of the TCP/IP subsystem (no default value). **For OCS:** Use the default value of TCPIP.	TCPIP	
VENDOR	The vendor value for IBM TCP/IP, Interlink TCP/IP, or OCS.	IBM	
SERVICE	The TCP/IP port number on which the server listens for inbound communications. You can specify a DNS name.		
HOST	The IP address of the machine on which the server runs. You can specify a DNS name.		

Reference Hub: Inbound LU6.2 Parameters Worksheet

Parameter	Description	Default Value	Supply Your Value Here
CLIENTLU	The VTAM resource name (APPLID) for the LU6.2 test client. This APPLID must be unique.		
SERVERLU	The VTAM resource name (APPLID) on which the server listens for LU6.2 inbound communications. This APPLID must be unique.		
MODENAME	The MODENAME with which the server's LU6.2 APPLID was defined.		

Reference Hub: Outbound TCP/IP Parameters Worksheet

Parameter	Description	Default Value	Supply Your Value Here
NODENAME	The name of the service on the subserver to which this server connects. If the subserver is on MVS, this value must match the name of a service specified in the subserver's EDASERVE file. You need to specify a node name for each outbound communications block. This name must be unique for each block. If the subserver is on UNIX or NT, this value must be a 1-8 character descriptive name.		
TCPNAME	**For IBM TCP/IP:** The name of the started task or the JOB name for the TCP/IP subsystem address space, if it is different from the default (TCPIP). **For Interlink TCP/IP:** The name of the TCP/IP subsystem (no default value). **For OCS:** Use the default value of TCPIP.	TCPIP	
VENDOR	The vendor value for IBM TCP/IP, Interlink TCP/IP, or OCS.	IBM	
SUBHOST	The IP address of the machine on which the subserver is running. You can specify a DNS name.		
SUBPORT	The port name or number on which the subserver is listening. You can specify a DNS name.		

Reference Hub: Outbound LU6.2 Parameters Worksheet

Parameter	Description	Default Value	Supply Your Value Here
LOCALLU	The server's LU6.2 APPLID. The server uses this APPLID to act like a client and communicate with the subserver. This keyword applies to SNA outbound communications only. If many clients connect to the same server, you may want to set this value to (LU6POOL).		
PARTLU	The LU6.2 APPLID on which the subserver listens for inbound communications.		
MODENAME	The MODENAME with which the subserver's LU6.2 APPLID was defined.		

Full-Function Server Configuration Worksheet

If you are configuring a Full-Function Server:

- Complete all relevant sections of the Full-Function Server Configuration Worksheet.
- After you complete the worksheets, you must install the data adapter before you generate the server. See Chapter 3, *Configuring Data Adapters*, to install the data adapter for this server.
- When you complete the data adapter installation, proceed to Chapter 4, *Generating a Server*. Use the values in your worksheet to edit *qualif*.EDALIB.DATA(EDACFGF). This member creates the configuration files for the Full-Function Server, with or without Hub Services.
- If you use FOCUS data sources, you must install the IBI subsystem. See Chapter 5, *Configuring Server System Features*, for more information.

Reference Full-Function: Configuration Parameters Worksheet

Parameter	Description	Default Value	Supply Your Value Here
qualif	The high–level qualifier for the data sets.		
OWNER	The Owner ID for the Dynamic Catalog views and tables. It is usually the TSO user ID of the person configuring the server.		
DUNIT	The disk unit used when allocating the data sets.		
SERVTYPE	The server type. Do not change this value.	FFS	FFS
LICENSEK	The license number supplied with this product on the insert.		
MAXIMUM	The maximum number of users for this Full-Function Server. It should not exceed the license agreement.		
DEPLOY	The deployment type for this server. Values can be PRIVATE, ISOLATED, or POOLED.	PRIVATE	
SYSOUT	The SYSOUT class the server uses for trace files.		

Reference Full-Function: Console Parameters Worksheet (Optional)

Parameter	Description	Default Value	Supply Your Value Here
CONAPPLD	The VTAM resource name (APPLID) used for Console access.		

Reference Full-Function: IMS DBCTL Parameters Worksheet (Optional)

Parameter	Description	Default Value	Supply Your Value Here
IMSRESLB	The DSN of the IMSRESLB library to be used by the server.		
IMSPZPLB	The DSN of the DFSPZP library in which the PZP load module has been generated.		
PZPSUFIX	The two-character suffix of the PZP module that is to be loaded from the DFSPZP library.		

Reference Full-Function: Inbound TCP/IP Parameters Worksheet

Parameter	Description	Default Value	Supply Your Value Here
TCPNAME	**For IBM TCP/IP:** The name of the started task or the JOB name for the TCP/IP subsystem address space, if it is different from the default (TCPIP). **For Interlink TCP/IP:** The name of the TCP/IP subsystem (no default value). **For OCS:** Use the default value of TCPIP.	TCPIP	
VENDOR	The vendor value for IBM TCP/IP, Interlink TCP/IP, or OCS.	IBM	
SERVICE	The TCP/IP port number on which the server listens for inbound communications. You can specify a DNS name.		
HOST	The IP address of the machine on which the server runs. You can specify a DNS name.		

Reference Full-Function: Inbound LU6.2 Parameters Worksheet

Parameter	Description	Default Value	Supply Your Value Here
CLIENTLU	The VTAM resource name (APPLID) for the LU6.2 test client. This APPLID must be unique.		
SERVERLU	The VTAM resource name (APPLID) on which the server listens for LU6.2 inbound communications. This APPLID must be unique.		
MODENAME	The MODENAME with which the server's LU6.2 APPLID was defined.		

Reference Full-Function: Outbound TCP/IP Parameters Worksheet (Optional)

Parameter	Description	Default Value	Supply Your Value Here
NODENAME	The name of the service on the subserver to which this server connects. If the subserver is on MVS, this value must match the name of a service specified in the subserver's EDASERVE file. You must specify a nodename for each outbound communications block. This name must be unique for each block. If the subserver is on UNIX or NT, this value must be a 1-8 character descriptive name.		
TCPNAME	**For IBM TCP/IP:** The name of the started task or the JOB name for the TCP/IP subsystem address space, if it is different from the default (TCPIP). **For Interlink TCP/IP:** The name of the TCP/IP subsystem (no default value). **For OCS:** The default value of TCPIP.	TCPIP	

Parameter	Description	Default Value	Supply Your Value Here
VENDOR	The vendor value for IBM TCP/IP, Interlink TCP/IP, or OCS.	IBM	
SUBHOST	The IP address of the machine on which the subserver is running. You can specify a DNS name.		
SUBPORT	The port name or number on which the subserver is listening. You can specify a DNS name.		

Reference **Full-Function: Outbound LU6.2 Parameters Worksheet (Optional)**

Parameter	Description	Default Value	Supply Your Value Here
LOCALLU	The server's LU6.2 APPLID. The server uses this APPLID to act like a client and communicate with the subserver. This keyword applies to SNA outbound communications only. If many clients connect to the same server, you may want to set this value to (LU6POOL).		
PARTLU	The LU6.2 APPLID on which the subserver listens for inbound communications.		
MODENAME	The MODENAME with which the subserver's LU6.2 APPLID was defined.		

Creating Network Definitions

Before you configure a server, the system administrator at your site must define the required system resources, such as APPLIDs for VTAM or port numbers for TCP/IP.

For sample network definitions for LU6.2 and TCP/IP, see *Installing VTAM Components for LU6.2* on page 2-15 and *Installing Components for IBM or Interlink TCP/IP* on page 2-16. Refer to IBM's VTAM documentation for a description of the parameters used in these sample network definitions.

Example **Installing VTAM Components for LU6.2**

If your application requires access to the server using an LU6.2 protocol, you must define two VTAM APPLIDs for use by the server.

The following is a sample of the required APPLID definitions:

```
*
* SYS1.VTAMLIST entries for LU6.2 protocol
*

ED38APP1        APPC=YES,           X
                DSESLIM=100,        X
                DMINWNL=50,         X
                DMINWNR=50,         X
                MODETAB=EDAMODE,    X
                DLOGMOD=LU62,       X
                VPACING=3,          X
                PARSESS=YES

ED38APP2        APPC=YES,           X
                DSESLIM=100,        X
                DMINWNL=50,         X
                DMINWNR=50,         X
                MODETAB=EDAMODE,    X
                DLOGMOD=LU62,       X
                VPACING=3,          X
                PARSESS=YES
```

Note:

- In the above example, ED38APP1 and ED38APP2 are LU6.2 APPLIDs used by the server for communication with Connector applications.
- The MODETAB specification must be included to refer to the log mode table entry required to support LU6.2 operation.

The following is an example of an LU6.2 log mode table entry that defines the communications characteristics of the VTAM APPLID definitions required by the server to support LU6.2 client connections:

```
EDAMODE    CSECT
EDAMODE    MODETAB
PARALLEL   MODEENT LOGMODE=PARALLEL,                                  X
               PSNDPAC=X'02',      PRIMARY SEND PACING COUNT          X
               SRCVPAC=X'02',      SECONDARY RECEIVE PACING CT        X
               SSNDPAC=X'02',      SECONDARY SEND PACING COUNT        X
               TYPE=0,             NEGOTIATED BIND                    X
               FMPROF=X'13',       FM PROFILE 19 LU 6.2               X
               TSPROF=X'07',       TS PROFILE 7 LU 6.2                X
               PRIPROT=X'B0',      PRIMARY NAU PORT                   X
               SECPROT=X'B0',      SECONDARY NAU PORT                 X
               RUSIZES=X'8989',                                       X
               COMPROT=X'78A5',    COMMON NAU PROT                    X
               PSERVIC=X'060200000000000000002F00'
```

Example Installing Components for IBM or Interlink TCP/IP

If your configuration consists of a Connector application connecting to a server using TCP/IP, you must define a service port. The service port can be limited to a single job name by adding an entry in the vendor-supplied communications file for TCP/IP. The file includes a section in which ports are defined.

Add the following lines to that section:

```
PORT
EDASERVE                            3000/TCP
```

The definition will prevent other jobs from accessing the port.

On the server, the Internet address is programmed into your TCP/IP communications processor (either the 3172 Interconnect Controller or the IBM 8232 LAN Channel Station). Your mainframe TCP/IP System Administrator should define the service designation. It should be unique to your server and entered through the SERVICE keyword in the TCP/IP communications file used by the server.

Users of IBM TCP/IP need to know the started task name for TCP/IP. Users of Interlink TCP/IP need to know the ACSS subsystem name.

CHAPTER 3

Configuring Data Adapters

Topics:

- Data Adapter Configuration Considerations
- Generating the Data Adapter for ADABAS
- Generating the Data Adapter for CA-DATACOM/DB
- Generating the Data Adapter for CA-IDMS/DB
- Generating the Data Adapter for DB2
- Generating the Data Adapter for IDMS/SQL
- Customizing the IMS/BMP Environment
- Customizing the IMS/DBCTL Environment
- Configuring the Data Adapter for InfoMan
- Generating the Data Adapter for Millennium
- Generating the Data Adapter for MODEL 204
- Generating the Data Adapter for Nomad
- Generating the Data Adapter for Oracle
- Generating the Data Adapter for SUPRA
- Generating the Data Adapter for SYSTEM 2000
- Generating the Data Adapter for Teradata

After selecting your server type, you must generate the data adapters needed for your site. Generating data adapters installs and customizes the data adapter for the server environment.

Depending on the data adapter you are installing, this may be a single step process, or it may require various types of environment customization or configuration.

Data Adapter Configuration Considerations

If you are configuring more than one instance of the server on the same machine, you need to repeat the entire configuration process for each instance. For example, if you have a Hub and a Full-Function Server, you need to follow the entire process for one and then the other. Alternatively, if you use the same installation library, *qualif*.EDALIB.LOAD, you only need to generate the data adapter once.

If you are accessing FOCUS, FUSION, or VSAM data sources, the data adapters are installed by default. For more information, see Chapter 4, *Generating a Server*.

Note: All assembly steps contained in the supplied sample JCL refer to the IEV90 program. If this program is not available at your site, an alias for IEV90 should be created pointing to program ASMA90.

Generating the Data Adapter for ADABAS

Edit and run *qualif*.EDALIB.DATA(GENEADL) to re-link ADALNK into FADALINK, which will be called by the server. The following is the JCL to be edited:

```
//*********************************************************************
//* Name:     GENEADL
//*
//* Function: Re-link ADALNK into FADALINK to be called with the
//*           ADABAS SVC number.
//* Substitutions: - Change "qualif" into the high-level qualifier
//*                  for your data sets.
//*                - Change "adabas.LOAD" to the name of the Software AG
//*                  supplied Adabas load library.
//*
//* Note:          - If you are running ADABAS 5.1.8 or earlier you must
//*                  change the mode card in the LKED step below to:
//*                        MODE AMODE(24),RMODE(24)
//*
//*********************************************************************
//*
//LKED     EXEC PGM=IEWL,PARM='NOLIST,NOXREF,LET'
//OLDMOD   DD   DISP=SHR,DSN=adabas.LOAD
//SYSLMOD  DD   DISP=SHR,DSN=qualif.EDALIB.LOAD
//SYSUT1   DD   UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   *
 INCLUDE OLDMOD(ADALNK)
 MODE AMODE(31),RMODE(ANY)
 ENTRY ADABAS
 NAME FADALINK(R)
//*
//*****************************************
```

```
//* TO ZAP FADALINK TO MAKE IT RE-ENTRANT  *
//* REVIEW THE ADALNK (OR ADALNK) SOURCE   *
//* SUPPLIED BY SOFTWARE AG. CHECK ZAP     *
//* ADDRESSES IN THE SOURCE CODE.          *
//* THIS STEP IS OPTIONAL.                 *
//*****************************************
//*ZAP      EXEC PGM=AMASPZAP
//*SYSLMOD  DD DISP=SHR,DSN=qualif.EDALIB.LOAD
//*SYSPRINT DD SYSOUT=*
//*SYSIN    DD *
//*  NAME FADALINK ADABAS
//*  VER  06  41E090A0
//*  REP  06  58E01018
//*
```

where:

adabas.LOAD

Is the name of the ADABAS load library supplied by Software AG.

qualif

Is the high-level qualifier for the data sets.

Generating the Data Adapter for CA-DATACOM/DB

To access DATACOM/DB tables, the server uses the load module DATACOM, located in EDALIB.LOAD, for all DATACOM communications. This load module is fully executable and does not require any installation steps. However, you must create certain components to access data using the Data Adapter for CA-DATACOM/DB.

The Data Adapter for CA-DATACOM/DB, like any other DATACOM application, uses a DATACOM User Requirements Table (URT) to access the appropriate DATACOM tables. A User Requirements Table contains all the tables that both the front-end application and the server itself can access. The server uses an Access File to specify a URT and a corresponding Master File to describe the table or tables accessed by the DATACOM URT.

The Access File contains the name of the URT that dynamically calls the DATACOM URT. For every URT that is specified in the Access File, there is a corresponding DATACOM URT. You can describe one DATACOM URT containing many DATACOM tables in one or more Master Files.

The URT may already exist for other DATACOM programs. If the DATACOM URT does not exist, or if you need to create a new DATACOM URT, see *How to Create a DATACOM URT* on page 3-4. You then must create a URT for each DATACOM URT. See *How to Create a Server-Supplied URT* on page 3-5 for more information.

Syntax How to Create a DATACOM URT

The following sample JCL, which you can use to create a DATACOM URT, is located in EDALIB.DATA in the member DTCMURT1.

```
//* * * * * * * JOB CARD GOES HERE * * * * * * * *
//*
//* DTCMURT1 - AUTODTCMb DATACOM TABLE URT CREATION
//*
//*  RUN THIS JOB BEFORE DTCMURT2, THIS WILL CREATE
//*  LOAD MEMBER AUTOURT1 IN LIBRARY ALLOCATED TO SYSLMOD
//*
//*  MODIFICATION OF THESE TABLES WILL ADVERSELY AFFECT AUTODTCM
//*
//ASMBL    EXEC  PGM=IEV90,PARM='OBJECT,NODECK,LIST',REGION=1024K
//SYSLIB   DD DSN=DATACOM.MACLIB,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=HIGHLVL.TEMPLIB(URT),DISP=SHR
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3   DD DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPUNCH DD DUMMY
//*SYSPUNCH DD UNIT=SYSDA,DISP=(,PASS),
//*            SPACE=(80,(100,100),RLSE),
//*            DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
//SYSIN    DD *
         DBURSTR                                                X
               ABEND=YES,                                       X
               MULTUSE=YES,                                     X
               PRTY=7
         DBURTBL                                                X
               TBLNAM={TableID},                                X
               DBID={nnn},                                      X
               SYNONYM=NO,                                      X
               ACCESS=RANSEQ,                                   X
               ELMCHG=NO,                                       X
               SEQBUFS=2,                                       X
               UPDATE=YES
         .
         .
         .
         DBUREND                                                X
               USRINFO=DATACOM URT
         END
/*
//LINKEDIT EXEC  PGM=IEWL,PARM='LET,NCAL,LIST,MAP,SIZE=1024K'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(10,1))
```

```
//TEMPLIB  DD DSN=HIGHLVL.TEMPLIB(URT),DISP=SHR
//SYSLMOD  DD DSN=HIGHLVL.TEST.LOADLIB,DISP=SHR
//SYSLIN   DD *
  INCLUDE TEMPLIB
  NAME    {LoadName}(R)
/*
```

where:

TableID

Is the DATACOM table's three-byte name.

nnn

Is the DATACOM table's three-position database number.

LoadName

Is the load name that will be dynamically called by the server-supplied URT.

Note: To add tables to a DATACOM URT, copy the DBURTBL block of parameters and change the following parameters. Do not change any other parameter values.

```
DBURTBL                                                          X
      TBLNAM={TableID},                                          X
      DBID={nnn},                                                X
.
.
.
NAME  {LoadName}(R)
```

Syntax

How to Create a Server-Supplied URT

To create a server-supplied URT, edit and run the DTCMURT2 job. Executing the DTCMURT2 is the only requirement to allow the server to access DATACOM. The output of this job is the server-supplied URT.

```
//* * * * * * * JOB CARD GOES HERE * * * * * * * *
/*
//* DTCMURT2 – SERVER-SUPPLIED URT CREATION
//*
//*  RUN THIS JOB AFTER DTCMURT1, THIS WILL CREATE A SERVER-SUPPLIED URT
//*
//ASMBL    EXEC  PGM=IEV90,PARM='OBJECT,NODECK,LIST',REGION=1024K
//SYSLIB   DD DSN=DATACOM.MACLIB,DISP=SHR
//         DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=HIGHLVL.TEMPLIB(URT),DISP=SHR
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3   DD DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPUNCH DD DUMMY
```

```
         DBURINF                                                    X
               URTABLE=LOAD,                                        X
               OPEN=USER,                                           X
               USRNTRY=USNTRY,                                      X
               LOADNAM={LoadName}
DBUREND                                                             X
   USRINFO=SERVER-SUPPLIED URT TO BE SPECIFIED IN ACCESS FILE DESCRIPTION
   END
/*
//LINKEDIT EXEC  PGM=IEWL,PARM='LET,NCAL,LIST,MAP,SIZE=1024K'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//OLDMOD   DD DSN=HIGHLVL.DATACOM.DATA,DISP=SHR
//TEMPLIB  DD DSN=HIGHLVL.TEMPLIB(URT),DISP=SHR
//SYSLMOD  DD DSN=HIGHLVL.TEST.LOADLIB,DISP=SHR
//SYSLIN   DD *
 INCLUDE OLDMOD(USNTRY)
 INCLUDE TEMPLIB
 ENTRY USNTRY
 NAME {Name}(R)
/*
```

where:

LoadName

Is the name of the DATACOM URT that is dynamically loaded by the server-supplied URT. It is identical to the LOADNAME parameter in job DTCMURT1. You may also use existing URT's by specifying their name here.

Name

Is the name of the server-supplied URT output that you create. It is any allowable 8-character name. This name is also described in the Access File. The corresponding Master File will contain element and field descriptions of one or more of the tables in the DATACOM URT.

Note: Do not modify DBURINF parameters.

Generating the Data Adapter for CA-IDMS/DB

The data adapter supports Release 10.x, 12.x, and 14.x of CA-IDMS/DB, and can operate in Central Version or Local Mode. See *Central Version Access for the Data Adapter for CA-IDMS/DB* on page 3-7 and *Local Mode Access for the Data Adapter for CA-IDMS/DB* on page 3-8.

Using your site criteria, allocate a library for your Access Files as ddname ACCESS. This library (or libraries) can be a mirror image of ddname MASTER and should have the same DCB parameters. A suggested name for this data set is:

```
qualif.EDAAFD.DATA
```

These allocations apply to the JCL:

- The library containing the Access Files for CA-IDMS/DB must be allocated to ddname ACCESS.
- If your server is running APF-authorized and your CA-IDMS/DB run-time load libraries are also APF-authorized, you must allocate them to STEPLIB. If your CA-IDMS/DB run-time load libraries are not authorized, allocate ddname NONAPFLB to the unauthorized CA-IDMS/DB libraries. Refer to the APFAUTH and TASKLIB keywords in the *iWay Server Administration for MVS and VM* manual.
- For Release 12.x and above, ddname SYSIDMS is required. Check with your CA-IDMS/DB DBA regarding this ddname.

Reference

Central Version Access for the Data Adapter for CA-IDMS/DB

You must allocate the ddname SYSCTL to the SYSCTL data set corresponding to the CV desired.

The CA-IDMS/DB functions will take place in the CA-IDMS/DB CV address space.

The subschema load modules will be searched in the following order:

1. CV primary load area (DDLDCLOD area).
2. CDMSLIB ddname from the CA-IDMS/DB CV job.
3. STEPLIB ddname from the CA-IDMS/DB CV job.

Reference Local Mode Access for the Data Adapter for CA-IDMS/DB

You must allocate all of the CA-IDMS/DB data sources to their respective ddnames. These files are assigned in the CA-IDMS/DB schema. You must only allocate the files associated with the subschema that you will be using.

All journal file allocations must be allocated along with the default Local Mode journal, ddname SYSJRNL, and all assigned to DD DUMMY. For example:

```
J1JRNL        DD DUMMY
SYSJRNL       DD DUMMY
```

Link-Editing User-Written Programs for the Data Adapter for IDMS

The Data Adapter for IDMS can execute user-written programs using the ZBIND and ZREADY exit facilities. See *Implementing the ZBIND Exit From the Data Adapter for IDMS* on page 3-8 and *Implementing the ZREADY Exit From the Data Adapter for IDMS* on page 3-12.

These programs can be written in any CA-IDMS/DB-supported language and must be link-edited to the Data Adapter for IDMS load module IDMSR.

The user-written programs initiated by the ZBIND and ZREADY exits must be link-edited as AMODE(24) and RMODE(24).

The Data Adapter for IDMS passes back a 4-byte STATUS, which is checked. An error condition occurs when this STATUS code is not equal to zero.

Implementing the ZBIND Exit From the Data Adapter for IDMS

The user-written program that is link-edited with the IDMSR module ZBIND is called using standard IBM calling conventions.

When you issue a request against a CA-IDMS/DB data source, the Data Adapter for IDMS calls the ZBIND exit before issuing the IDMS BIND RUN UNIT command.

The following parameters are passed to the ZBIND exit from the Data Adapter for IDMS. Upon return from the exit, the last parameter contains an IDMS error status. The return codes are shown in the COBOL program that follows.

Call ZBIND with the following parameters:

```
USER        PIC X(08)
SUBSCHEMA   PIC X(08)
DBNAME      PIC X(08)
NODE        PIC X(08)
DICTNAME    PIC X(08)
DICTNODE    PIC X(08)
STATUS      PIC X(04)
```

On return of a non-zero STATUS code, you will receive the message:

```
EDA4405 BIND RUN-UNIT DENIED BY USER EXIT ZBIND FOR SUBSCHEMA: SUBSCHEMA
```

Example Implementing the ZBIND Exit

The following is a sample COBOL program located in *qualif*.EDALIB.DATA(ZBINDC) that implements one possible function of the ZBIND exit. This program verifies that a user has authorization to access the subschema that is associated with a request.

```
*RETRIEVAL
NO-ACTIVITY-LOG
IDENTIFICATION DIVISION.
PROGRAM-ID. ZBIND.
*********************************************************
* PROGRAM: ZBIND
* PURPOSE: CHECK TO SEE IF A USER ISSUING A REQUEST
* HAS ACCESS TO THE SUBSCHEMA REQUESTED.
* THIS PROGRAM MAY BE USED FREELY BY THOSE
* WHO NEED AN EXAMPLE TO CODE THEIR OWN EXIT.
*********************************************************
DATE-WRITTEN. JAN  1996.
DATE-COMPILED.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
SPECIAL-NAMES.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
IDMS-CONTROL SECTION.
PROTOCOL. MODE IS BATCH DEBUG IDMS-RECORDS MANUAL.
SCHEMA SECTION.
DB IDMSNWKA WITHIN IDMSNTWK.
DATA DIVISION.
    EJECT
FILE SECTION.
    EJECT
WORKING-STORAGE SECTION.
01  WS-START.
    02 FILLER              PIC X(33) VALUE
                          'ZBIND WORKING STG STARTS HERE==>'.
01  COPY IDMS SUBSCHEMA-CONTROL.
01  COPY IDMS RECORD USER-047.
01  COPY IDMS RECORD ACCESS-045.
01  COPY IDMS RECORD SS-026.
LINKAGE SECTION.
01  PUSER       PIC X(8).
01  PSS         PIC X(8).
01  PDBNAME     PIC X(8).
01  PNODE       PIC X(8).
01  PDICT       PIC X(8).
01  PDICTNODE   PIC X(8).
```

```
01  PSTAT       PIC X(4).
*************************************************************
*
* RETURN CODE - 0000 - USER HAS ACCESS TO SUBSCHEMA
*             - 0016 - USER NOT IN DICTIONARY
*             - 0032 - USER DOES NOT HAVE ACCESS TO SUBSCHEMA
*             - 9999 - PASSED PARMS USER AND/OR SS INVALID
*             - NNNN - ERROR DURING IDMS-STATS PROCESSING.
*                      WHERE NNNN = ERROR-STATUS
*
*************************************************************
 PROCEDURE DIVISION  USING PUSER,
                           PSS,
                           PDBNAME,
                           PNODE,
                           PDICT,
                           PDICTNODE,
                           PSTAT.
 0000-MAINLINE.

***  DISPLAY '"'  PUSER '"'
***          '"'  PSS   '"'.

*** RESET ERROR-STATUS TO ENSURE REENTRANCY
     MOVE '1400' TO ERROR-STATUS.
     MOVE 'ZBIND' TO PROGRAM-NAME.
*** CHECK FOR INVALID PASSED PARMS
     IF PUSER = SPACE OR
        PSS   = SPACE
         MOVE '9999' TO PSTAT
         GOBACK.
*** CHECK FOR PRESENCE OF DBNAME PARM - DETERMINE WHICH BIND
     NOTE: THIS PROGRAM ASSUMES THE DBNAME PASSED IS THE DBNAME
     WHICH MAPS THE IDMSNWKA SUBSCHEMA TO THE REQUESTED DICTIONARY.
     IT DOES ==> NOT <== LOOK AT THE DICTNAME PARAMETER PASSED TO
     THE EXIT.
     IF PDBNAME = SPACE
         BIND RUN-UNIT
       ELSE
         BIND RUN-UNIT  DBNAME PDBNAME.
     IF ANY-ERROR-STATUS
         MOVE ERROR-STATUS TO PSTAT
         GOBACK.
*** BIND AND READY
     BIND USER-047.
     IF ANY-ERROR-STATUS
         MOVE ERROR-STATUS TO PSTAT
         GO TO 9999-WRAP-UP.
```

```
     BIND ACCESS-045.
     IF ANY-ERROR-STATUS
         MOVE ERROR-STATUS TO PSTAT
         GO TO 9999-WRAP-UP.
BIND SS-026.
     IF ANY-ERROR-STATUS
         MOVE ERROR-STATUS TO PSTAT
         GO TO 9999-WRAP-UP.
     READY USAGE-MODE IS RETRIEVAL
     IF ANY-ERROR-STATUS
         MOVE ERROR-STATUS TO PSTAT
         GO TO 9999-WRAP-UP.
*** SEE IF USER IS IN DICTIONARY
    MOVE PUSER TO USER-NAME-047.
     FIND CALC USER-047.
     IF DB-REC-NOT-FOUND
         MOVE '0016' TO PSTAT
         GO TO 9999-WRAP-UP.
     IF ANY-ERROR-STATUS
         MOVE ERROR-STATUS TO PSTAT
         GO TO 9999-WRAP-UP.

*** LOOP THRU USER-ACCESS-SS RECS TO SEE IF RELATIONSHIP EXISTS
 1200-NEXT-ACCESS.
     FIND NEXT ACCESS-045 WITHIN USER-ACCESS.
     IF DB-END-OF-SET
         MOVE '0032' TO PSTAT
         GO TO 9999-WRAP-UP.
     IF ANY-ERROR-STATUS
         MOVE ERROR-STATUS TO PSTAT
         GO TO 9999-WRAP-UP.
     IF NOT SS-ACCESS MEMBER
         GO TO 1200-NEXT-ACCESS.
     OBTAIN OWNER WITHIN SS-ACCESS.
     IF ANY-ERROR-STATUS
         MOVE ERROR-STATUS TO PSTAT
         GO TO 9999-WRAP-UP.
     IF SS-NAM-026 = PSS
         MOVE '0000' TO PSTAT
         GO TO 9999-WRAP-UP.
     GO TO 1200-NEXT-ACCESS.

 9999-WRAP-UP.
     FINISH.
     GOBACK.
     EJECT
*COPY IDMS IDMS-STATUS.
*IDMS-ABORT SECTION.
*I-A -EXIT. EXIT.
```

You can use the following JCL to link-edit the ZBIND user-written program to the Data Adapter for IDMS load module IDMSR. This link JCL is located in *qualif*.EDALIB.DATA(ZBIND).

```
//LINK     EXEC PGM=IEWL,
//         PARM='LET,NCAL,SIZE=(1024k),LIST'
//OBJLIB        DD DSN=qualif.zbind.objlib,DISP=SHR
//LOAD          DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//SYSLMOD       DD DSN=qualif.OUTPUT.LOADLIB,DISP=SHR
//SYSUT1        DD UNIT=SYSDA,SPACE=(CYL,(10,1))
//SYSPRINT      DD SYSOUT=*
//SYSLIN        DD  *
  INCLUDE OBJLIB(ZBIND)
  INCLUDE LOAD(IDMSR)
  ENTRY IDMSR
  NAME IDMSR(R)
/*
//
```

where:

qualif.zbind.objlib

Is the library containing the ZBIND object code.

qualif.EDALIB.LOAD

Is the library containing the IDMSR load module.

qualif.OUTPUT.LOADLIB

Is the library containing the new IDMSR load module.

Implementing the ZREADY Exit From the Data Adapter for IDMS

The user-written program that is link-edited with the IDMSR module ZREADY is called using standard IBM calling conventions.

When you issue a request against a CA-IDMS/DB data source, the Data Adapter for IDMS calls the ZREADY exit before issuing an IDMS READY command.

The Data Adapter for IDMS calls the ZREADY exit for each request against a CA-IDMS/DB data source, once for each area that contains fields that are referenced in the request. This enables the site to take action based on the USERID, subschema, and area, to either allow the area to be readied or terminate the request. Like the ZBIND exit, the application program determines whether processing will continue. A non-zero value in STATUS will cause processing to stop. If ZREADY indicates that READY is not to be issued, the server informs you and terminates that particular request without retrieving any data. This occurs even if the areas were previously readied, following which it will end its contact with CA-IDMS/DB by issuing a FINISH command. Note that the session itself is not ended, and that you are free to try to access other IDMS areas.

The following parameters are passed to the ZREADY exit from the Data Adapter for IDMS. Upon return from the exit, the last parameter contains an IDMS error status. The return codes are shown previously in the sample COBOL program.

Call ZREADY with the following parameters:

```
SUBSCHEMA   PIC X(08)
AREANAME    PIC X(32)
DBNAME      PIC X(08)
NODE        PIC X(08)
DICTNAME    PIC X(08)
DICTNODE    PIC X(08)
STATUS      PIC X(04)
```

On return of a non-zero STATUS code, you will receive the following message:

```
EDA4382 READY INHIBITED BY USER EXIT ZREADY FOR AREA:AREANAME
```

Example Implementing the ZREADY Exit

You can use the following JCL to link-edit the ZREADY user-written program to the Data Adapter for IDMS load module IDMSR. This link JCL can be found in *qualif*.EDALIB.DATA(ZREADY).

```
//LINK     EXEC PGM=IEWL,
//         PARM='LET,NCAL,SIZE=(1024k),LIST'
//OBJLIB        DD DSN=qualif.zready.objlib,DISP=SHR
//LOAD          DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//SYSLMOD       DD DSN=qualif.OUTPUT.LOADLIB,DISP=SHR
//SYSUT1        DD UNIT=SYSDA,SPACE=(CYL,(10,1))
//SYSPRINT      DD SYSOUT=*
//SYSLIN        DD  *
  INCLUDE OBJLIB(ZREADY)
  INCLUDE LOAD(IDMSR)
  ENTRY IDMSR
  NAME IDMSR(R)
/*
//
```

where:

qualif.zready.objlib

Is the library containing the ZREADY object code.

qualif.EDALIB.LOAD

Is the library containing the IDMSR load module.

qualif.OUTPUT.LOADLIB

Is the library containing the new IDMSR module.

Generating the Data Adapter for DB2

Edit and run *qualif*.EDALIB.DATA(RELJINS) or (RE3JINS) depending on your version of OS/390 and z/OS. See the *iWay Server Installation* manual for more information.

The tablespace specified for TBSPC is used as the default run-time tablespace. If an existing tablespace is not available, a new one must be created before performing this step. Default space parameters can be used when you are creating a tablespace.

Edit the following JCL, add a job card, and submit the job.

```
//EDAPROCS JCLLIB ORDER=qualif.EDALIB.DATA
//RELGDB2  EXEC RELQGDB2,
//         PREFIX='qualif',
//         ATTACH=CAF,
//         AUTO=auto,
//         DSNLOAD='db2.load.library',
//         DSNMACS='db2.macro.library',
//         ER=er,
//         NCRS=ncrs,
//         NSTMTS=nstmts,
//         PLAN='edaplan',
//         PLANOWN=planownr,
//         SSID='ssid',
//         TBSPC='catalog-tablespace   may-not-be-blank',
//         WORK=work
```

where:

qualif

Is the high-level qualifier for the data sets.

auto

Is the run-time default for thread autoclose mode. Possible values are:

`0` for AUTOCLOSE OFF.

`1` for AUTOCLOSE ON.

db2.load.library

Is the name of the DB2 load library.

db2.macro.library

Is the name of the DB2 macro library.

er

Is the run-time default for error message mode. Possible values are:

`1` for server messages.

`2` for server and DB2 messages.

`ncrs`

Is the maximum number of concurrent select cursors available at run-time. To specify a single digit number, you must include a zero as a first digit. For example, NCRS=08 would specify 8 concurrent select cursors. To specify a 3-digit number, you must modify the "MVS00" parameter in *qualif*.EDALIB.DATA(RELQGDB2). See *Specifying an NCRS and NSTMTS Number Greater Than 99* on page 3-16 for more information.

`nstmts`

Is the maximum number of concurrently allocated INSERT, DELETE, and UPDATE statements. To specify a single digit number, you must include a zero as a first digit. For example, NSTMTS=08 would specify a maximum number of 8 concurrently allocated statements. To specify a 3-digit number, you must modify the "MVS00" parameter in *qualif*.EDALIB.DATA(RELQGDB2). See *Specifying an NCRS and NSTMTS Number Greater Than 99* on page 3-16 for more information.

`edaplan`

Is the run-time default for the application plan name. Do not change the total number of places (8) between single quotation marks when altering this parameter. Pad the parameter with trailing blanks if needed.

`planownr`

Is the authorization ID of the bound plan.

`ssid`

Is the run-time default for the DB2 subsystem ID. The default value supplied is DSN. Do not change the total number of places (4) between single quotation marks when altering this parameter. Pad the parameter with trailing blanks if needed.

`catalog tablespace`

Is the run-time default for the DB2 database.tablespace. Do not change the total number of places (36) between single quotation marks when altering this parameter. Pad the parameter with trailing blanks if needed.

`work`

Specifies a DASD work unit. It defaults to SYSDA.

Note: Do not change the total number of spaces between quotation marks for PLAN, SSID, or TBSPC when substituting the parameters. Pad the parameter with trailing blanks if needed.

Example Editing the JCL for the Data Adapter for DB2

The following is an example of the JCL that has been edited for the Data Adapter for DB2:

```
//EDAPROCS JCLLIB ORDER=EDA.V5R1M00.EDALIB.DATA
//RELGDB2  EXEC RELQGDB2,
//         PREFIX='EDA.V5R1M00',
//         ATTACH=CAF,
//         AUTO=0,
//         DSNLOAD='DSN610.DSNLOAD',
//         DSNMACS='DSN610.DSNMACS',
//         ER=1,
//         NCRS=80,
//         NSTMTS=16,
//         PLAN='EDAPLAN',
//         PLANOWN=EDADBA,
//         SSID='DSN ',
//         TBSPC='EDADBA.CATALOG',
//         WORK=SYSDA
```

Specifying an NCRS and NSTMTS Number Greater Than 99

You must modify the "MVS00" parameter in member *qualif*.EDALIB.DATA(RELQGDB2) as shown below:

```
//   ('SYSPARM(&FUNCT&NCRS&NSTMTS&SSID&PLAN&TBSPC&AUTO&IM&ER.MVS00)')
```

The last two digits of the MVS00 parameter will determine resulting values. The first digit will determine the number of hundreds added to NCRS and the second number of hundreds added to NSTMTS. For example, MVS11 and defaults NCRS=80 and NSTMTS=16 will create an interface capable of 180 cursors and 116 statements.

Note: Each cursor adds 632 bytes and each statement adds 328 bytes, so 100 additional cursors and 100 additional statements will add 96K.

Granting Access to DB2

After the RELJINS or RE3JINS member name has been run, you should grant execute privileges on the plan to all users who will be using this data adapter. If you plan to run the server with user-level security, see *Customizing the DB2 Security Exit* on page 3-17. Using SPUFI or another DB2 tool, issue the command

```
GRANT EXECUTE ON PLAN edaplan TO PUBLIC;
```

where:

edaplan

Is the value that was defined by the parameter PLAN= in member *qualif*.EDALIB.DATA(RELJINS) or (RE3JINS).

The following is an example of a command issued to grant execute privileges:

```
GRANT EXECUTE ON PLAN EDAPLAN TO PUBLIC;
```

An RDBMS such as DB2 requires that all programmed interaction with the data source be controlled at the program module level. The program is represented to the data source using an object called a *plan*.

The installation procedure automatically creates a plan for the server. When the server accesses the RDBMS, it uses that plan name. The RDBMS checks the user's authorization ID (usually the sign-on user ID) against the plan name to determine whether the user has authority to access the data source with that plan.

Any user accessing the RDBMS through the server must be granted authority to execute the plan. For more information on plans, see the applicable IBM DB2 manuals.

If you need to customize the DB2 security exit, see *Customizing the DB2 Security Exit* on page 3-17.

Customizing the DB2 Security Exit

You must customize the DB2 security exit in order to allow the Adapter to run with user-level security enabled. If this is not done, all users will be assigned the connect ID to DB2 that is associated with the region, job submitter, or started task.

If these changes are made, each user will connect to DB2 with the authorization of the user ID with which they logged onto the server. The server must also be running with security turned on. For instructions and examples, see *Changing the DSN3SATH Exit for RACF and CA-TOP SECRET* on page 3-17 and *Changing the DSN3SATH Exit for CA-ACF2* on page 3-20.

Example Changing the DSN3SATH Exit for RACF and CA-TOP SECRET

The following example shows the changes to be made to the IBM DB2 signon exit, DSN3SATH, which should be used for RACF and CA-TOP SECRET sites.

The arrows shown in the following code indicate the lines containing the recommended modification of DSN3SATH, which calls the module, FOCDSN3, the supplied exit.

Once you have finished the edits, assemble the exit into an object file. This object file is input to the link JCL found in *Modifying the Link JCL for DSN3SATH* on page 3-21.

Note:

- The positioning of these lines is appropriate, assuming that no other changes or additions have already been made to DSN3SATH. If any changes have been made, decide on the most appropriate location for this call to FOCDSN3.
- FOCDSN3 is used to set the proper primary (individual user ID) authorization.

- Another program, FOCDSN4, is used to set the proper secondary (group ID) authorization for RACF and CA-TOP SECRET. FOCDSN4 is not needed with CA-ACF2; the secondary authorization ID(s) will be set correctly without it.
- The source code for the FOCDSN3 and FOCDSN4 programs is found in *qualif*.EDALIB.DATA(FOCDSN3S) and *qualif*.EDALIB.DATA(FOCDSN4S), respectively.

Changes in DSN3SATH for RACF (Version 6.1 example code) and CA-TOP SECRET:

```
SATH001  DS    0H
         USING WORKAREA,R11        ESTABLISH DATA AREA ADDRESSABILITY
         ST    R2,FREMFLAG         SAVE FREEMAIN INDICATOR
         XC    SAVEAREA(72),SAVEAREA CLEAR REGISTER SAVE AREA
         LA    R15,SAVEAREA        GET ADDRESS OF CSECT'S SAVE AREA
         ST    R13,FOUR(,R15)      CHAIN THE SAVE AREA BACK POINTER
         ST    R15,EIGHT(,R13)     CHAIN SAVEAREA FORWARD
         LR    R13,R15             ADDRESS OF CSECT'S SAVE AREA SPACE
         XC    EXPLARC,EXPLARC     INIT RETURN CODE TO NORMAL RETURN
         XC    SECCOUNT,SECCOUNT   CLEAR GROUP NAME COUNTER FIELD
         L     R8,PSAAOLD-PSA      GET CURRENT ASCB ADDRESS AND
                                   USING ASCB,R8 SET MAPPING
                                   ADDRESSABILITY

         EJECT
**********SECTION 1:  DETERMINE THE PRIMARY AUTHORIZATION ID  *********
*                                                                     *
*   IF THE INPUT AUTHID IS NULL OR BLANKS, CHANGE IT TO THE AUTHID    *
*   IN EITHER THE JCT OR THE FIELD POINTED TO BY ASCBJBNS.            *
*                                                                     *
*   THE CODE IN THIS SECTION IS AN ASSEMBLER LANGUAGE VERSION OF      *
*   THE DEFAULT IDENTIFY AUTHORIZATION EXIT. IT IS EXECUTED ONLY      *
*   IF THE FIELD ASXBUSER IS NULL UPON RETURN FROM THE RACROUTE       *
*   SERVICE. FOR EXAMPLE, IT DETERMINES THE PRIMARY AUTH ID FOR       *
*   ENVIRONMENTS WITH NO SECURITY SYSTEM INSTALLED AND ACTIVE.        *
*                                                                     *
***********************************************************************
         SPACE
--->     LA    R1,AIDLPRIM         LOAD PARM REG1
--->     CALL  FOCDSN3             GO GET EXIT
         CLI   AIDLPRIM,BLANK      IS THE INPUT PRIMARY AUTHID NULL
         BH    SATH020             SKIP IF A PRIMARY AUTH ID EXISTS
         L     R7,ASCBCSCB         GET CSCB ADDRESS
         CLI   CHTRKID-CHAIN(R7),CHTSID IS IT TSO FOREGROUND ADDR SPACE
         BNE   SATH010             BRANCH IF NOT
         L     R7,ASCBJBNS         GET ADDRESS OF LOGON ID
         MVC   AIDLPRIM,0(R7)      MAKE IT THE PRIMARY AUTH ID
         B     SATH019             TO END OF THIS ROUTINE
SATH010  DS    0H                  NOT TSO, BUT BATCH OR STC SPACE
         L     R6,PSATOLD-PSA      CURRENT TCB ADDRESS
```

```
         L     R7,TCBJSCB-TCB(,R6) CURRENT JSCB ADDRESS
         L     R5,JSCBJCT-IEZJSCB(,R7) CURRENT JCT ADDRESS
         LA    R5,X'10'(,R5)       ADJUST FOR CORRECT DSECT MAPPING
         CLI   JCTUSER-INJMJCT(R5),X'4E' IF JCTUSER PLUS SIGN OR LESS
         BNH   SATH019             THEN LEAVE AIDLPRIM BLANK KEB0026
         MVC   AIDLPRIM(7),JCTUSER-INJMJCT(R5) COPY JOB USER ID
         MVI   AIDLPRIM+7,BLANK    ASSURE BLANK PADDING
SATH019  DS    0H                  END OF ROUTINE
         EJECT
*****SECTION 2:  DETERMINE THE LIST OF SECONDARY AUTHORIZATION IDS*****
*                                                                     *
*    THIS SECTION IS WRITTEN SPECIFICALLY FOR THE RACF ENVIRONMENT.   *
*    IT CAN/SHOULD BE REPLACED FOR OTHER SECURITY PRODUCTS.           *
*                                                                     *
***********************************************************************
*                                                                     *
*    IF RACF IS ACTIVE AND THE LIST OF GROUPS OPTION IS ALSO ACTIVE,  *
*    USE THE CGRP AREA TO GET THE CONNECTED GROUP NAMES.              *
*    COPY THEM TO THE SECONDARY ID LIST IN THE AIDL.                  *
*                                                                     *
***********************************************************************
         SPACE
         CLI   AIDLPRIM,BLANK      IS THE INPUT PRIMARY AUTHID NULL
         BNH   SATH090             EXIT IF PRIMARY AUTH ID NULL
SATH020  DS    0H                  BRANCH TO HERE IF PRIMARY EXISTS


*****OPTIONAL CHANGE @CHAR7:  FALLBACK TO SEVEN CHAR PRIMARY AUTHID****
*                                                                     *
*    IF YOUR INSTALLATION REQUIRES ONLY SEVEN CHARACTER PRIMARY       *
*    AUTHORIZATION IDS (POSSIBLY TRUNCATED) DUE TO DB2 PRIVILEGES     *
*    GRANTED TO TRUNCATED AUTHORIZATION IDS, THEN YOU MUST BLANK OUT  *
*    COLUMN 1 OF THE ASSEMBLER STATEMENT IMMEDIATELY FOLLOWING THIS   *
*    BLOCK COMMENT. THEN ASSEMBLE THIS PROGRAM AND LINK-EDIT IT INTO  *
*    THE APPROPRIATE DB2 LOAD LIBRARY AS EXPLAINED IN AN APPENDIX     *
*    OF "THE DB2 ADMINISTRATION GUIDE".                               *
*                                                                     *
*    OTHERWISE, YOU NEED DO NOTHING.                                  *
*                                                            @KYD0271*
***********************************************************************
*        MVI   AIDLPRIM+7,BLANK    BLANK OUT EIGHTH CHARACTER    @CHAR7
         SPACE
         L     R5,CVTPTR           ADDRESS MVS CVT
         L     R7,CVTRAC-CVT(,R5)  RACF CVT ADDRESS
         LTR   R7,R7               IF RACF CVT ADDRESS ZERO,
         BZ    SATH049             RACF IS NOT EVEN INSTALLED
         USING RCVT,R7             SET BASE FOR RACF CVT
         TM    RCVTSTAT,RCVTRNA    IS RACF ACTIVE
```

```
         BO    SATH049             SKIP AROUND IF NOT
         SPACE 1
*   RACF IS ACTIVE ON THIS MVS
         USING ACEE,R6             ESTABLISH BASE FOR ACEE     @KYL0108
         ICM   R6,B'1111',AIDLACEE CALLER PASSED ACEE ADDRESS? @KYL0108
         BZ    SATH024             NO, USE ADDRESS SPACE ACEE  @KYL0108
         CLC   ACEEACEE,EYEACEE    IS IT REALLY AN ACEE?       @KYL0108
         BE    SATH027             YES, PROCEED NORMALLY       @KYL0108
         SPACE 1
SATH024  DS    0H                  USE ADDRESS SPACE ACEE      @KYL0108
         L     R6,ASCBASXB         GET ADDRESS SPACE EXTENSION BLOCK
         L     R6,ASXBSENV-ASXB(,R6)  GET ACEE ADDRESS
--->     CALL  FOCDSN4             GO GET EXIT (4=GROUP AUTH)
         LTR   R6,R6               DOES AN ACEE EXIST? IF NOT,
         BZ    SATH049             SKIP AROUND CONNECTED GROUP NAME
         CLC   ACEEACEE,EYEACEE    DOES IT LOOK LIKE AN ACEE?
         BNE   SATH049             NO, THEN CAN'T DO GROUPS
         DROP  R8                  DROP ASCB BASE REG          @TU25003
         SPACE 1
SATH027  DS    0H                  CHECK LIST OF GROUPS OPTION @KYL0108
         TM    RCVTOPTX,RCVTLGRP   IS LIST OF GROUPS CHECKING ACTIVE
         BZ    SATH040             SKIP TO SINGLE GROUP COPY IF NOT
         DROP  R7                  DROP RCVT BASE REG          @TU25003
         SPACE 1
*   RACF LIST OF GROUPS OPTION IS ACTIVE
```

Example Changing the DSN3SATH Exit for CA-ACF2

This example illustrates changes in DSN3SATH for CA-ACF2:

```
*****************************************************************
*       PRIMARY AUTHORIZATION ID                                *
*****************************************************************
*
-->     LA      R1,AIDLPRIM           POINT TO AUTH FIELD
-->     CALL    FOCDSN3               CALL TASK-LEVEL-EXIT
        CLI     AIDLPRIM,C'  '        PRIMARY AUTHID THERE?
        BH      PRIMTSO               ..YES, EVERYTHING OK HERE
        L       R3,PSAAOLD-PSA(0)     CURRENT ASCB ADDRESS
```

Example Modifying the Link JCL for DSN3SATH

This is sample link JCL for the IBM exit DSN3SATH. Modify the JCL to link the modules into the DB2 security exit as follows:

```
//LKED  EXEC PGM=IEWL,PARM='LIST,XREF,LET,RENT,AMODE=31'
//OBJECT    DD DSN=db2pref.SDSNSAMP.OBJ,DISP=SHR <--OUTPUT OF ASSEMBLE
STEP
//EDAMOD    DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//SYSLMOD   DD DSN=db2pref.DSNEXIT,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(100,(50,50))
//SYSLIN    DD *
  INCLUDE OBJECT
  INCLUDE EDAMOD(FOCDSN3)
  INCLUDE EDAMOD(FOCDSN4)      <---OMIT FOR CA-ACF2
          ENTRY DSN3@ATH
  NAME DSN3@ATH(R)
/*
```

where:

db2pref

Is the prefix for the DB2 data sets.

qualif

Is the high-level qualifier for the data sets.

Once this job finishes successfully, you must recycle the DB2 subsystem for the changes to take effect.

Generating the Data Adapter for IDMS/SQL

Edit and run *qualif*.EDALIB.DATA(GENEIDQ) to link module IDQFOC, which is called by the server.

The following is the JCL to be edited.

```
//*********************************************************************
//* Name:     GENEIDQ
//*
//* Function: Linkedit the IDMS/SQL interface
//*
//* Substitutions:- Change "qualif" into the high-level qualifier for
your
//*                 data sets.
//*               - Change "idms.load" to the name of your IDMS load
//*                 library.
//*
//* Note:    When installing the Interface for use under CICS, the
//*          CA-IDMS Interface program name must be changed
//*          from "IDMS" which is used for non-CICS systems
//*          to "IDMSCINT" which is used for CICS. Change the
//*          indicated control statement to "INCLUDE IDMSLIB(IDMSCINT)"
//*********************************************************************//*
//LKED     EXEC PGM=IEWL,PARM='LIST,NOXREF,LET'
//IDMSLIB  DD   DISP=SHR,DSN=idms.load
//SYSLMOD  DD   DISP=SHR,DSN=qualif.EDALIB.LOAD
//SYSUT1   DD   UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   *
  INCLUDE SYSLMOD(IDQFOC)
  INCLUDE IDMSLIB(IDMS)                <- see note above
  MODE AMODE(31),RMODE(ANY)
  ENTRY IDQFOC
  NAME IDQFOC(R)
//*
```

where:

idms.load

Is the name of the IDMS/SQL load library.

qualif

Is the high-level qualifier for the data sets.

Customizing the Server JCL for IDMS/SQL

If your server is running APF-authorized and your IDMS/SQL run-time load libraries are also APF-authorized, you must allocate them to STEPLIB. If your IDMS/SQL run-time load libraries are not authorized, allocate ddname NONAPFLB to the unauthorized IDMS/SQL libraries and add the following keywords to the global section of the EDASERVE configuration file:

```
ADFAUTH=INTERNAL
TASKLIB=NONAPFLIB
```

Allocating Data Sets for CV Access to IDMS/SQL

You must allocate the ddname SYSCTL to the SYSCTL data set corresponding to the CV desired. The IDMS/SQL functions will take place in the IDMS/SQL CV address space.

The subschema load modules are located and retrieved in the following order:

1. CV primary load area (DDLDCLOD area).
2. CDMSLIB ddname from the IDMS/SQL CV job.
3. STEPLIB ddname from the IDMS/SQL CV job.

Note: When data is accessed from secondary dictionaries, the primary load area is not searched to retrieve the subschema/dmcl modules.

The dmcl used is the CV's global dmcl, and it may or may not be the same as the dmcl associated with the subschema.

Allocating IDMS/SQL Data Sources for Local Mode Access

You must allocate all of the IDMS/SQL data sources to their respective ddnames, which are assigned in the IDMS/SQL schema.

All journal file allocations must be made available along with the default Local Mode journal, ddname SYSJRNL, assigned to DD DUMMY.

Customizing the IMS/BMP Environment

Refer to this topic if *read only* access to IMS data sources is required, but IMS/DBCTL is *not* installed on your system. Currently, SQL update of IMS data sources through the IMS/BMP environment is not available.

The XMI Server is an application program that intercepts DL/I calls from the server and issues them to IMS. Before you can invoke the data adapter with this configuration, the IBI Subsystem *must* be installed on the server (see Chapter 5, *Configuring Server System Features*), and there must be an XMI Server job executing with the appropriate parameter settings for your configuration. Also, the XMI Server must have a PCB available for the data sources you want to access.

Accessing the XMI Server

Accessing the XMI Server is a two-part process:

1. Initiating the XMI Server.

   To initiate the XMI Server, run a region controller program and pass to it parameters describing the environment you need. The following chart lists the most common parameters:

Parameter	Value	Definition
Mode	BMP DLI	Indicates Batch Message Processing mode. This mode accesses online IMS data sources through the IMS/TM address space. For details, see *Initiating the XMI Server in BMP Mode* on page 3-26. Indicates local mode. This mode accesses IMS data sources that you allocate locally. For details, see *Initiating the XMI Server in DLI Mode* on page 3-27.
Program	XMI	Loads and accesses the XMI Server in the region controller address space.
PSB	psbname	Identifies the PSB to use.

   One XMI Server can work for multiple users, but each user must have a separate PCB. Therefore, to allow multiple users, the PSB for the server should contain duplicate PCBs.

2. Invoking the data adapter once an XMI Server job is running.

   XMI Server programs can be started at any time. In normal usage, one such job will be started in the morning with additional jobs started throughout the day, as the need for additional PCBs arises.

   The job class and time expiration parameters used with the XMI Server job must allow for a long residence in the system, and the priority should be sufficiently high in order to give good response to the users being serviced. In setting the various job parameters that affect response, remember that these jobs spend most of their time in the WAIT state consuming no resources, since they are idle when not responding to DL/I calls. Even when processing these calls, they only do minimal work. All the logical work of the data adapter is done on the server.

   When you use the XMI Server to access IMS data sources, you must allocate a common communications file (to ddname FOCBMP) in both the server and the XMI Server address space. See *Allocating the FOCBMP Communications File* on page 3-25.

   For related information, see *Invoking the Server in the XMI Server Environment* on page 3-30.

Allocating the FOCBMP Communications File

The XMI Server runs in a different address space from the server and the Data Adapter for IMS/DB . The two address spaces must establish a link before they can communicate with one another. A communications file assists this handshake function by acting as a bulletin board, notifying the server of the XMI Server's whereabouts. After the handshake is completed, the communications file plays no role in subsequent message traffic.

You can initiate up to 16 concurrent XMI Server jobs, each of which requires its own communications file.

You must now allocate as many communications files as needed, up to 16. Data set names are irrelevant, but for clarity, you can use the names '*qualif*.EDABMP.DATA', '*qualif*.EDABMP1.DATA',…, '*qualif*.EDABMP15.DATA'. You can make the initial allocations through the IEFBR14 utility or the TSO ALLOCATE command, provided that it can catalog data sets on permanently mounted disk packs. For an illustration, see *Allocating Communications Files Using the IEFBR14 Utility* on page 3-25.

After allocating the FOCBMP communications file, you must decide which mode to run the XMI Server in:

- For BMP mode, see *Initiating the XMI Server in BMP Mode* on page 3-26.
- For DLI mode, see *Initiating the XMI Server in DLI Mode* on page 3-27.

Example Allocating Communications Files Using the IEFBR14 Utility

The following example illustrates the allocation of two communications files using the IEFBR14 utility

```
//STEP1 EXEC PGM=IEFBR14
//DD1   DD   DSN=qualif.EDABMP.DATA,
//      DISP=(NEW,CATLG),DCB=BLKSIZE=16,
//      SPACE=(TRK,(l)),VOL=SER=volser
//DD2   DD   DSN=qualif.EDABMP1.DATA,
//      DISP=(NEW,CATLG),DCB=BLKSIZE=16,
//      SPACE=(TRK,(l)),VOL=SER=volser
```

where:

qualif

Is the high-level qualifier for the data sets.

volser

Is a valid volume ID for your site.

Note that the minimum possible amount of space is allocated. Users must have read access to the communications files. The communications file is also discussed in *Allocating the Common Communications File for the XMI Server* on page 3-31.

Initiating the XMI Server in BMP Mode

BMP (Batch Message Processing) mode accesses online IMS data sources under the control of IMS/TM. Therefore, you can access Fast Path databases with this configuration, and, if a record is updated, you retrieve the updated version.

To initiate an XMI Server with this configuration, submit the following JCL after editing it to conform to your site's standards and adding a JOB card.

```
//BMPXMI EXEC
   PGM=DFSRRC00,REGION=reg,PARM='BMP,XMI,psbname[,,,,,,,,,nba,oba]'
//STEPLIB     DD DSN=IMS.RESLIB,DISP=SHR
//            DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//FOCBMP      DD DSN=qualif.EDABMP.DATA,DISP=SHR
//FOCPSB      DD DSN=qualif.EDAPSB(psbname),DISP=SHR
//DFSRESLB    DD DSN=IMS.RESLIB,DISP=SHR
//ERRORS      DD DSN=qualif.EDAMSG.DATA,DISP=SHR
//SYSOUT      DD SYSOUT=*
//
```

where:

reg

Is the region size.

psbname

Is the name of the PSB to use.

nba

Is the number of buffers for the normal buffer allocation. It is required only for access to Fast Path databases. Omit it if you will not access Fast Path.

Note: There are nine commas between the *psbname* and *nba* parameters.

oba

Is the number of buffers for the overflow buffer allocation. It is required only for access to Fast Path databases. Omit it if you will not access Fast Path.

If you omit the *nba* and *oba* parameters and attempt to access Fast Path databases, the following message is generated:

```
(EDA4214) REMOTE DLI CALL ERROR STATUS FOR SEGMENT:XXX/ FR
```

FR is an IMS status code indicating that too few buffers were allocated.

qualif

Is the high-level qualifier for the data sets.

Note:

- IMS.RESLIB is the IMS load library.
- The ddname FOCBMP is allocated to the common communications file. You must allocate the same file when you invoke the server. See *Invoking the Server in the XMI Server Environment* on page 3-30 for details.
- Most error conditions detected by the XMI Server are reflected back to the originating application. The SYSOUT file allocated in the XMI Server job will contain only global error messages pertaining to the XMI Server environment as a whole, such as open errors on ddname FOCBMP or FOCPSB.
- PCBs for the XMI Server are discussed in *Creating a PSB for the XMI Server* on page 3-32.

Once the XMI Server job is executing, you can invoke the server from JCL. For details, see *Invoking the Server in the XMI Server Environment* on page 3-30.

After initiating the XMI Server in BMP mode, see *Initiating the XMI Server With Data Sources Under Control of CICS* on page 3-29.

Initiating the XMI Server in DLI Mode

DLI mode accesses IMS data sources that you allocate locally in the DFSRRC00 address space. Since you do not access the IMS/TM address space, you cannot use Fast Path databases and, if a record is updated during your session, you cannot retrieve the updated record.

Note: If you substitute the parameter DBB for DLI, IMS uses ACBs when accessing data sources. This ensures that the data source is in synch with the IMS descriptions you are using.

To initiate an XMI Server with this configuration, submit the following JCL after editing it to conform to your site's standards and adding a JOB card. Note that you must allocate your IMS data sources, DBDs, and PSBs in this JCL.

You do not have to submit this JCL if an XMI Server job with an available PCB is already running with the following parameter settings

```
//DLIXMI    EXEC PGM=DFSRRC00,REGION=region,PARM='DLI,XMI,psbname'
//STEPLIB     DD DSN=IMS.RESLIB,DISP=SHR
//            DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//DFSRESLB    DD DSN=IMS.RESLIB,DISP=SHR
//IMS         DD DSN=IMS.PSBLIB,DISP=SHR
//            DD DSN=IMS.DBDLIB,DISP=SHR
//DFSVSAMP    DD DSN=IMS.DFSVSAMP(bufpool),DISP=SHR
//PATDB01     DD DSN=IMS.PATIENT.DB,DISP=SHR
//PATDBIX     DD DSN=IMS.PATIENT.IX,DISP=SHR
//PATDBIX1    DD DSN=IMS.PATIENT.IX1,DISP=SHR
//PATDBIX2    DD DSN=IMS.PATIENT.IX2,DISP=SHR
//PATDBIX3    DD DSN=IMS.PATIENT.IX3,DISP=SHR
//FOCBMP      DD DSN=qualif.EDABMP.DATA,DISP=SHR
//FOCPSB      DD DSN=qualif.IMS.EDAPSB(psbname),DISP=SHR
//ERRORS      DD DSN=qualif.EDAMSG.DATA,DISP=SHR
//SYSOUT      DD SYSOUT=*
//
```

where:

region

Is the region size.

psbname

Is the name of the PSB to use.

qualif

Is the high-level qualifier for the data sets.

bufpool

Is the member that contains your VSAM buffer pool information.

Note:

- The ddnames allocated to the IMS data sources are from the DD1 parameters in the relevant IMS DBDs.
- IMS.RESLIB is the IMS load library, IMS.DBDLIB is the IMS DBD library, and IMS.PSBLIB is the IMS PSB library.
- If you specify the parameter DBB instead of DLI on the EXEC card, you must include the ACB library in the allocation for ddname IMS.
- The ddname FOCBMP is allocated to the common communications file. You must allocate the same file when you invoke the server. See *Invoking the Server in the XMI Server Environment* on page 3-30 for details.

- Most error conditions detected by the XMI Server are reflected back to the originating application. The SYSOUT file allocated in the XMI Server job will contain only global error messages pertaining to the XMI Server environment as a whole, such as open errors on ddname FOCBMP or FOCPSB.
- PCBs for the XMI Server are discussed in *Creating a PSB for the XMI Server* on page 3-32.

Once the XMI Server job is executing, you can invoke the server job. See *Invoking the Server in the XMI Server Environment* on page 3-30 for a description.

After initiating the XMI Server in BMP mode, see *Initiating the XMI Server With Data Sources Under Control of CICS* on page 3-29.

Initiating the XMI Server With Data Sources Under Control of CICS

Program DFHDRP accesses online IMS data sources under the control of CICS. This configuration consumes a lot of communications resources and, since there is no IMS/TM address space, it cannot access Fast Path databases. However, since the data sources are online, if a record is updated, you retrieve the updated version.

Note:

- You should use this configuration only when CICS controls the data sources in non-share mode. If your site has DBRC and sharing is allowed, use the configuration described in *Initiating the XMI Server in BMP Mode* on page 3-26.
- IMS Version 5.1 does not support this configuration. You must use DBCTL instead.
- If you are running against a CICS system where the DLI system is AMODE 24, you must run the JCL found in *qualif*.EDALIB.DATA(XMIDLI24) before you use the XMI Server to CICS for IMS access.

To initiate an XMI Server with this configuration, submit the following JCL after editing it to conform to your site's standards and adding a JOB card.

You do not have to submit this JCL if an XMI Server job that has an available PCB is already running with the following parameter settings

```
//XMICICS EXEC
   PGM=DFHDRP,REGION=region,PARM='SSA,PGM=XMI,psbname,cics_applid'
//STEPLIB    DD DSN=CICS.LOAD,DISP=SHR
//           DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//DFHLIB     DD DSN=CICS.LOAD,DISP=SHR
//FOCBMP     DD DSN=qualif.EDABMP.DATA,DISP=SHR
//FOCPSB     DD DSN=qualif.EDAPSB(psbname),DISP=SHR
//ERRORS     DD DSN=qualif.EDAMSG.DATA,DISP=SHR
//SYSOUT     DD SYSOUT=*
//
```

where:

region

Is the region size.

psbname

Is the name of the PSB to use.

cics_applid

Is the VTAM application ID for the CICS address space you will access.

qualif

Is the high-level qualifier for the data sets.

Note:

- CICS.LOAD is the CICS load library.
- The ddname FOCBMP is allocated to the common communications file. You must allocate the same file when you invoke the server, as described in *Invoking the Server in the XMI Server Environment* on page 3-30.
- Most error conditions detected by the XMI Server are reflected back to the originating application. The SYSOUT file allocated in the XMI Server job will contain only global error messages pertaining to the XMI Server environment as a whole, such as open errors on ddname FOCBMP or FOCPSB.
- PCBs for the XMI Server are discussed in *Creating a PSB for the XMI Server* on page 3-32.

Once the XMI Server job is executing, you can invoke the server from JCL, as described in *Invoking the Server in the XMI Server Environment* on page 3-30.

Invoking the Server in the XMI Server Environment

Before you can invoke the server in the XMI Server environment, an XMI Server job, initiated with the appropriate parameter settings, must be running and must have an available PCB for each data source you will access. For more information about PCBs for the XMI Server, see *Creating a PSB for the XMI Server* on page 3-32.

Include the following allocation in your server JCL

```
//FOCBMP    DD DSN=qualif.EDABMP.DATA,DISP=SHR
```

where:

qualif

Is the high-level qualifier for the data sets.

Allocating the Common Communications File for the XMI Server

You must allocate the common execution files needed for the XMI Server. The "handshake" between the data adapter, which is running in the server, and XMI, which is running in another address space, requires a communications file accessible to both. This file must be allocated to ddname FOCBMP. It consists of a single 80-byte record, and it plays no role in subsequent message traffic between the two programs. Space for this file must be allocated once (on a permanently mounted disk pack) and cataloged as part of the installation of the XMI Server. The data set name is irrelevant.

Note: This discussion assumes that the server production libraries are stored in data sets cataloged under the same high-level qualifier. The identifier *qualif* denotes this common high-level qualifier.

After the communications file has been created, when you subsequently invoke the server and the XMI Server, allocate this file as follows:

- In the JCL that initiates the XMI Server:

  ```
  //FOCBMP DD DSN=qualif.EDABMP.DATA,DISP=SHR
  ```

- In the application:

From	Allocation
Server JCL	`//FOCBMP DD DSN=qualif.EDABMP.DATA,DISP=SHR`
EDASPROF or user profile	`DYNAM ALLOC FILE FOCBMP DATASET qualif.EDABMP.DATA SHR`

When several concurrent XMI jobs are active, each one requires its own communications file. Each XMI Server job must allocate a different file—it does not matter which one—to ddname FOCBMP. For example:

XMI job 1:

```
//FOCBMP DD DSN=qualif.EDABMP.DATA,DISP=SHR
```

XMI job 2:

```
//FOCBMP DD DSN=qualif.EDABMP1.DATA,DISP=SHR
```

The server must allocate *all* of the communications files corresponding to the XMI jobs it might want to use. Each communications file must be allocated to a different but standard ddname. These ddnames are FOCBMP, and FOCBMP1 through FOCBMP15. For example:

```
//FOCBMP   DD  DSN=qualif.EDABMP.DATA,DISP=SHR
//FOCBMP1  DD  DSN=qualif.EDABMP1.DATA,DISP=SHR
```

If none of the FOCBMP files are allocated, the server does not attempt to use the XMI Server, but will use DBCTL access if it is active.

Creating a PSB for the XMI Server

Since IMS uses the PCB to maintain an application program's position in the data source, concurrent users who are serviced by the same XMI job cannot share PCBs.

Each user application that accesses the server queries all the XMI jobs to which it is linked, through the communications files, until it finds one that has a free PCB appropriate for the SQL query on the data source. (The search for an available PCB is transparent to the application.) The PCB remains occupied for the duration of the retrieval portion of a request, and is freed prior to the output portion of the request.

Thus, if ten users need concurrent access to the SALES data source, the PSB must contain at least ten PCBs that correspond to the Master File for SALES. The eleventh user receives a diagnostic message indicating that there is no free PCB available at the time. (In this case, the user should initiate another XMI Server, as illustrated in *Allocating the Common Communications File for the XMI Server* on page 3-31.) The ten SALES PCBs can reside:

- In a single PSB that requires only one XMI job.
- In ten identical PSBs with one SALES PCB each for ten concurrent XMI jobs.
- In any other combination that adds up to ten.

The IMS DBA must determine which configuration is best for the site. The trade-off is between a single large PSB that ties up only one XMI region or several XMI regions, each running a non-duplicated and therefore smaller PSB.

Note: If the PSB contains multiple PCBs for the same data source, you must duplicate the whole block of PCBs in order to allow access through secondary indexes.

Terminating an XMI Server

A successfully started XMI Server job will not end automatically. The operator must terminate it in one of three ways:

- **By executing job XMISTOP.** This is the recommended termination method. For details, see *How to Terminate an XMI Server With XMISTOP* on page 3-33.
- **Through an IMS cancel.** The IMS cancel will properly disconnect the XMI Server and IMS/TM regions without any impact on the latter.
- **Through an MVS cancel.** The MVS cancel should be avoided because of its potential impact on the IMS/TM region if the cancellation occurs while the XMI Server is processing a DL/I call.

Termination through job XMISTOP avoids problems. The job sends the server a termination message that breaks into the server message queue ahead of all other messages addressed to it. Upon receipt of this message, the server performs its cleanup and returns to IMS, signaling a normal completion.

You can invoke any of the three termination methods at any time, without impact on those who are actively using the canceled job (except for the impact on IMS/TM discussed previously). All such users will receive one of several error messages, depending on the point during the retrieval process when the cancellation occurred.

Syntax

How to Terminate an XMI Server With XMISTOP

The following is sample JCL for the XMISTOP job, which is based on the member *qualif*.EDACTL.DATA(EDADEBUG)

```
//*********************************************************
//**         Sample EDADEBUG Script                      **
//*********************************************************
//*
//* Supply a proper job card.
//*
//XMISTOP    XEC PGM=TSCOM3
//STEPLIB    DD  DISP=SHR,DSN=qualif.EDALIB.LOAD
//ERRORS     DD  DISP=SHR,DSN=qualif.EDAMSG.DATA
//FOCEXEC    DD  DISP=SHR,DSN=qualif.EDARPC.DATA
//FOCBMP     DD  DISP=SHR,DSN=comfile
//*IBITROUT  DD  SYSOUT=*,DCB=(LRECL=132,BLKSIZE=132,RECFM=F)
//*IBITRACE  DD  DISP=SHR,DSN=qualif.EDACTL.DATA(IBITRACE)
//*********************************************************
//**         TSCOM3 Script Input and Output Files        **
//*********************************************************
//TRMOUT     DD  SYSOUT=*,DCB=(LRECL=133,BLKSIZE=133,RECFM=F)
//TRMIN      DD  *
EX XMI ACTION=STOP,GATEWAY=FOCBMP
/*
```

where:

qualif

Is the high-level qualifier for the data sets.

comfile

Is the data set name of the communications file that is allocated to the XMI Server job you want to terminate (for example, *qualif*.EDABMP.DATA).

Customizing the IMS/DBCTL Environment

Refer to this topic only if you need to install the IMS/DBCTL access to IMS data. To complete the IMS/DBCTL configuration, see *Configuring IMS/DBCTL Access* in Chapter 8, *Modifying the Server Configuration*.

Important: Do not refer to Chapter 8, *Modifying the Server Configuration*, until *after* you have completed the IMS/DBCTL installation steps.

Before installing the data adapter, you should verify that the site is running IMS Version 3.1 or higher.

Make sure all APPLCTN macros describing PSBs that will be accessed by multiple users specify SCHDTYP=Parallel. If necessary, modify the APPLCTN macros for such PSBs to include the attribute SCHDTYP=Parallel, and re-run the IMS SYSGEN to make the changes effective.

The customization process includes these steps:

- Create the DRA startup table. See *How to Create the DRA Startup Table: DFSPZPxx* on page 3-34.
- Assemble and link the DRA startup table. See *How to Assemble and Link the DRA Startup Table* on page 3-35.
- Verify that the DBCTL is operational. See *How to Verify That the DBCTL Is Operational* on page 3-38.
- Create the FOCPSB Library. See *How to Create the FOCPSB Library (Mandatory for DBCTL Access)* on page 3-40.

You may also wish to establish security for the Data Adapter for IMS. See *Establishing Security* on page 3-40.

Syntax

How to Create the DRA Startup Table: DFSPZPxx

The Database Resource Adapter (DRA) is the interface between a user task and DBCTL. The DRA Startup Table contains values that define the characteristics of the DRA. The DRA Startup Table is named

`DFSPZP`*`xx`*

where:

xx

Is a two-character suffix that should be chosen to comply with your site's standards.

Once this suffix has been chosen, its value is included in the IMSPZP attribute in the DBCTL definition of the server configuration file (EDASERVE).

Proceed to *How to Assemble and Link the DRA Startup Table* on page 3-35.

Syntax

How to Assemble and Link the DRA Startup Table

After you have created the DRA Startup Table by specifying parameters in the DFSPRP macro of the DFSPZP*xx* module, you must assemble and link the DFSPZP*xx* module into the *qualif*.EDALIB.LOAD library as illustrated in the following sample JCL.

The sample JCL illustrates how to specify the parameters. You can normally find a sample JCL in your IMS installation library. For a list of required keywords and their descriptions, see *IMS Keywords* on page 3-36.

```
//job card goes here
//ASSEMBLE EXEC  PGM=IEV90,REGION=2M,PARM='OBJECT,NODECK'
//SYSLIB   DD  DSN=IMS.MACLIB,DISP=SHR
//SYSLIN   DD  UNIT=SYSDA,DISP=(,PASS),
//             SPACE=(80,(100,100),RLSE),
//             DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
//SYSPRINT DD  SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1   DD  UNIT=SYSDA,DISP=(,DELETE),
//             SPACE=(CYL,(10,5))
//SYSIN    DD  *
PRP      TITLE 'DATABASE RESOURCE ADAPTER STARTUP PARAMETER TABLE'
DFSPZPxx CSECT
         EJECT
         DFSPRP DSECT=NO,                                          X
                DBCTLID=IMS3,                                      X
                DDNAME=DFSRESLB,                                   X
                DSNAME=IMS.RESLIB,                                 X
                CNBA=150,                                          X
                MAXTHRD=150,                                       X
                MINTHRD=5
         END
/*
//*
//LINK   EXEC  PGM=IEWL,PARM='XREF,LIST',COND=(0,LT,ASSEMBLE),
//             REGION=4M
//SYSLIN   DD  DSN=*.ASSEMBLE.SYSLIN,DISP=(OLD,DELETE)
//SYSPRINT DD  SYSOUT=*,DCB=BLKSIZE=1089
//SYSUT1   DD  UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//             SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
//SYSLMOD  DD  DISP=SHR,DSN=qualif.EDALIB.LOAD(DFSPZPxx)
```

where:

qualif

Is the high-level qualifier for the data sets.

xx

Is the two-character suffix chosen for the DRA Startup Table.

Note: The *qualif*.EDALIB.LOAD library must be concatenated ahead of IMS.RESLIB in the allocation for ddname STEPLIB.

Proceed to *How to Verify That the DBCTL Is Operational* on page 3-38.

Reference IMS Keywords

The following chart describes the required keywords. Other keywords that affect your IMS environment and may be required at your site are described in the *IBM IMS/ESA Version 4 System Definition Reference*.

Keyword	Description	Default
AGN	Is a 1- to 8-character application group name used as part of the DBCTL security function. For more information on DBCTL security, see the *IMS/ESA System Administration Guide*.	N/A
CNBA	Is the total number of Fast Path buffers for the server's use. For a description of Fast Path DEDB buffer usage, see the *IMS/ESA System Administration Guide*. You can omit this parameter if your site does not utilize Fast Path.	N/A
DBCTLID	Is the 4-character name of the DBCTL region. This is the same as the IMSID parameter in the DBC procedure. For more information on the DBC procedure, see the *IBM IMS/ESA Version 3 System Definition Reference*.	SYS1
DDNAME	Must be DFSRESLB. It is the ddname that will be allocated to the DBCTL RESLIB library (see the DSNAME keyword).	N/A
DSNAME	Is the 1- to 44-character data set name of the DBCTL RESLIB library. It must contain the DRA modules and be MVS authorized.	IMS.RESLIB

Keyword	Description	Default
FPBOF	Is the number of Fast Path DEDB overflow buffers to be allocated per thread. For a description of Fast Path DEDB buffer usage, see the *IMS/ESA System Administration Guide*. You can omit this keyword if your site does not use Fast Path.	00
FPBUF	Is the number of Fast Path DEDB buffers to be allocated and fixed per thread. For a description of Fast Path DEDB buffer usage, see the *IMS/ESA System Administration Guide*. You can omit this parameter if your site does not utilize Fast Path.	00
FUNCLV	Is the level of the DRA that the CCTL supports. FUNCLV=1 means the CCTL uses the DRA at the IMS 3.1 level.	1
MAXTHRD	Is the maximum number of concurrent DRA threads. The value cannot exceed 255.	1
MINTHRD	Is the minimum number of concurrent DRA threads available. Since the number of threads specified by this keyword will be allocated at all times, regardless of whether they are used, be careful when selecting this value. The value cannot exceed 255.	1
SOD	Is the output class to use for a SNAP DUMP of abnormal thread termination.	A
TIMEOUT	Is the number of seconds a CCTL should wait for the successful completion of a DRA TERM request. Specify this value only if the CCTL is coded to use it. This value is returned to the CCTL upon completion of an INIT request.	60
TIMER	Is the number of seconds between attempts of the DRA to identify itself to DBCTL during an INIT request.	60
USERID	Is an 8-character name of the CCTL region. No two CCTLs (servers accessing the same IMS region through DBCTL) can have the same USERID (address space ID).	N/A

Syntax How to Verify That the DBCTL Is Operational

The JCL found in *qualif*.EDALIB.DATA(IMDEDA) verifies that the DBCTL is properly installed. Run this job after making any changes necessary to the JCL to suit your site's standards and naming conventions. This step should be run prior to installing the data adapter.

Note: If you are running IMS/ESA V3.1, make sure you have applied APAR PL70841.

```
//*********************************************************************
//* Name:     IMDEDA
//*
//* Function: TEST DBCTL ENVIRONMENT
//*
//* Date:     AUGUST 2001
//*
//* Substitutions:
//*
//*       qualif     - Change to the high-level qualifier
//*                    for your data sets.
//*
//*       edapsb1    - Change to any PSB name.
//*
//*       pcicspsb   - RACF class to be used in the test. If this
//*                    parameter is omitted, security checking will
//*                    not be performed.
//*
//*       xx         - Is the 2-character suffix that identifies the DRA
//*                    startup module.
//*
//*********************************************************************
//STEP1     EXEC PGM=IMDTEST,PARM='emppsb1,pcicspsb,xx',
//          REGION=4096K,TIME=(1,5)
//STEPLIB   DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//          DD DSN=IMS.RESLIB,DISP=SHR
//SYSIN     DD DUMMY
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
```

where:

emppsb1

Is a PSB name.

pcicspsb

Is a security class.

xx

Is a 2-character suffix that identifies the DRA Startup Module to be loaded, as described in *How to Create the DRA Startup Table: DFSPZPxx* on page 3-34.

Note: If you omit this parameter, its value defaults to 00, which may cause IMS to load the incorrect module.

qualif

Is the high-level qualifier for the data sets.

After running the JCL, examine the JES output. The return codes displayed on the following lines indicate that DBCTL is properly installed:

```
12.45.47 JOB05736  +CCTL: JOBNAME(USERID1) PSB(EMPPSB1) CLASS(PCICSPSB)
12.45.47 JOB05736  +CCTL: INIT CALLED
12.45.47 JOB05736  +CCTL: INIT RETURNED,RETC(00000000)
12.45.47 JOB05736  +CCTL: WAITING FOR CONTROL EXIT
12.45.47 JOB05736  +CTLX: CONTROL EXIT CALLED
12.45.47 JOB05736  +CTLX: CONTROL EXIT RETURNED
12.45.47 JOB05736  +CCTL: INIT COMPLETED, FUNC(02) SFNC(00) RCOD(00)
RETC(00000000)
12.45.47 JOB05736  +CCTL: SECURITY CHECK COMPLETED, RC(00000004)
12.45.47 JOB05736  +CCTL: TERMINATE DRA CALLED
12.45.47 JOB05736  +CTLS: SUSPEND EXIT CALLED
12.45.47 JOB05736  +CTLR: RESUME EXIT CALLED
12.45.47 JOB05736  +CTLS: SUSPEND EXIT RETURNED
12.45.47 JOB05736  +CTLR: RESUME EXIT RETURNED
12.45.47 JOB05736  +CCTL: TERMINATE DRA COMPLETED, RETC(00000000)
```

Note that the installation is successful even when the security check completes with a return code of 4.

If the JES output indicates an error, make sure that:

- A DRA Startup Table with a suffix of 00 exists. See *How to Create the DRA Startup Table: DFSPZPxx* on page 3-34.

  If it exists, check that:

  - The parameters specified in it (especially DSNAME and DDNAME) are correct.
  - The DBCTLID keyword points to the correct IMS system.
- DBCTL was installed at your site. If DBCTL was installed, check that it is up and running.

Proceed to *How to Create the FOCPSB Library (Mandatory for DBCTL Access)* on page 3-40.

Syntax How to Create the FOCPSB Library (Mandatory for DBCTL Access)

Create *qualif*.EDAPSB.DATA, which is the FOCPSB library. At run time, this library will be allocated to the ddname FOCPSB. Its members are IMS PSB descriptors used by the server to describe IMS PSB structures.

The following JCL shows the parameters used to create the FOCPSB library

```
//CREATE  EXEC PGM=IEFBR14
//DD1     DD DSN=qualif.EDAPSB.DATA,
//        DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(10,20,20)),
//        DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600),UNIT=sysda,
//        VOL=SER=volser
```

where:

qualif

Is the high-level qualifier for the data sets.

sysda

Is the default system DASD UNIT type.

volser

Is the volume ID on which you would like to allocate the library.

Establishing Security

Two types of security are available with the data adapter:

The data adapter provides the option of adding security restrictions and passwords to Master Files. The Data Administrator can encrypt Master Files to conceal the security information.

Note: Security and the ENCRYPT command are described in the *iWay Server Administration for MVS and VM* manual.

- The DBCTL environment, when accessed through the server, enables use of security systems through the standard SAF interface. With the SAF interface, your site can use security products such as RACF, CA-TOP SECRET, and CA-ACF2 to restrict access to PSBs. Before allowing access to a particular PSB, the security system verifies that the user is authorized to read the PSB.
- The object of the security feature is to ensure that users access only those PSBs for which they have authorization. The server can query standard security systems through the standard SAF interface before allowing a user to access a PSB.

The DBCTL function is tested and verified with the RACF product. Other SAF products using the identical calls should perform properly when installed and verified by your site's security administrator.

RACF comes with several predefined security classes. Customer sites can use an existing class (such as PCICSPSB) or define a resource class specifically for DBCTL use. For an illustration, see *Defining a PSB Resource With User Permissions* on page 3-41.

Example Defining a PSB Resource With User Permissions

The following example illustrates how to define a PSB resource through a PCICSPSB profile to RACF, and how to grant users permission to access the resource.

```
RDEFINE PCICSPSB (psbname) UACC(NONE) NOTIFY(sys_admin_userid)
PERMIT psbname CLASS(PCICSPSB) ID(user_or_group [user_or_group ...])
ACCESS(READ)
```

where:

psbname

Is a PSB name to be protected by RACF.

sys_admin_userid

Is the user ID of the system administrator.

user_or_group

Authorizes the user IDs and/or user groups listed in the PERMIT command to read the specified PSB. Separate items in the list with blanks.

At run time, after the PSB is selected but prior to scheduling it, the server issues a call to the security system and verifies that the user is authorized to read the PSB.

Configuring the Data Adapter for InfoMan

The Data Adapter for InfoMan is installed as part of the standard server installation procedure. Configure the data adapter after the distribution tape has been unloaded. For information regarding the installation procedure, refer to the *iWay Server Installation* manual.

The AUTOIMAN configuration file, IMANCONF, contains information relevant to the PIDT Selection Library, the Master File, and the Access File.

Note: If ddname FOCEXEC is allocated to concatenated partitioned data sets, IMANCONF must reside in the first partitioned data set.

Procedure How to Customize the IMANCONF Configuration File

The IMANCONF configuration file consists of three lines:

1. `INFOFVT.V6R1M0.SBLMFMT`
2. *`qualif`*`.MASTER.DATA`
3. *`qualif`*`.ACCESS.DATA`

Line 1 is the PIDT Selection Library. INFOFVT.V6R1M0.SBLMFMT is the default provided with Information/Management. You may keep this library as the default, or enter a different PIDT Selection Library name.

Lines 2 and 3 are the partitioned data sets (PDSs) for the Master and Access Files. In both lines, substitute your user ID for *qualif*. The PDS names can remain as shown, or be changed. Remember that only cataloged PDSs can be referenced.

Note: Only the first 35 characters of each of the lines will be read.

In order to maintain multiple IMANCONF configuration files, you must copy the IMANCONF configuration file into your own IMANCONF partitioned data set. Follow the editing instructions in *Editing the Server JCL for IBM Information/Management* on page 3-43 to customize the file.

The three lines of data that comprise the IMANCONF configuration file are the permanent AUTOIMAN defaults, and also appear on the AUTOIMAN window, Master File Generation Facility for InfoMan Main Menu. You can change the PIDT Selection Library and the Master and Access Files on the Main Menu. However, the changes you make on the Main Menu are effective only for the duration of the AUTOIMAN session, and do not become permanent AUTOIMAN defaults.

Example Editing the Server JCL for IBM Information/Management

The Data Adapter for InfoMan can be enabled in a server. The following is an annotated example of the server JCL containing the appropriate allocations for an IBM Information/Management data source.

```
//JOBCARD
//*****************************************************
//*
//*  Server JCL
//*
//*  with InfoMan Data Adapter libraries.
//*
//*****************************************************
//EDASERVE  EXEC PGM=SSCTL,
//STEPLIB    DD  DISP=SHR,DSN=qualif.EDALIB.LOAD
  .
  .
  .
//           DD  DISP=SHR,DSN=INFOFVT.V6R1M0.BLX1.LOAD
//           DD  DISP=SHR,DSN=INFOFVT.V6R1M0.VSAMLOAD
//           DD  DISP=SHR,DSN=INFOFVT.V6R1M0.FIXLOAD
//           DD  DISP=SHR,DSN=INFOFVT.V6R1M0.SBLMMOD1
//           DD  DISP=SHR,DSN=INFOFVT.V6R1M0.SESSLOAD
  .
  .
  .
//MASTER     DD  DISP=SHR,DSN=qualif.EDAMFD.DATA
//ACCESS     DD  DISP=SHR,DSN=qualif.EDAAFD.DATA
//*FSTRACE   DD  SYSOUT=*,DCB=(LRECL=132,BLKSIZE=132,RECFM=F)
//*FSTRACE4  DD  SYSOUT=*,DCB=(LRECL=132,BLKSIZE=132,RECFM=F)
```

where:

qualif

Is the high-level qualifier for the data sets.

Note:

- STEPLIB is the allocation for the load library, *qualif*.EDALIB.LOAD. Allocate this library before any IBM Information/Management data sets or other system load libraries.

  All libraries in STEPLIB must be APF-authorized in order for the server load modules to run.

  If your installation uses non-APF authorized libraries, you must allocate only *qualif*.EDALIB.LOAD (which must be APF-authorized) in STEPLIB, and all non-APF authorized libraries under ddname EDALIB.

- MASTER is the allocation for the *qualif*.MASTER.DATA data set, in which Master Files are found.

  The AUTOIMAN facility will create Master Files to describe IBM Information/Management data to the server.

- ACCESS is the allocation for the *qualif*.ACCESS.DATA data set, in which Access Files are found.

  The AUTOIMAN facility will create Access Files to describe IBM Information/Management data to the server.

- FSTRACE, if uncommented, logs all trace output from the Data Adapter for InfoMan. FSTRACE creates an entry for each retrieved record, and is a useful method for logging all communication with the IBM Information/Management data source.
- FSTRACE4, if uncommented, logs only trace output summaries. FSTRACE4 displays Program Interface Argument Table (PIAT) criteria, and the number of hits to the IBM Information/Management data source (that is, the number of records that satisfied the search criteria). It is effective in filtering out the additional output that FSTRACE provides.

PICA Parameters for the Data Adapter for InfoMan

The Data Adapter for InfoMan interfaces with the low-level InfoMan Application API. Since the low-level API acquires application parameters from the Program Interface Communications Area (PICA), the Data Adapter for InfoMan inputs certain pre-defined parameters to PICA.

PICA parameters of special interest to the Information/Management database administrators include:

- **PICAUSRN:** Application ID or User ID.
- **PICASESS:** Session Member Name.
- **PICACLSN:** Privilege Class Name.
- **PICADBID:** Database ID.
- **PICATINT:** Transaction Time Interval. See *How to Specify PICATINT in an InfoMan Access File* on page 3-45.

The Information/Management data source uses the application ID PICAUSRN in conjunction with the privilege class name PICACLSN to establish the level of user accessibility permitted by the data source.

If you specify a PICAUSRN (application ID) and/or PICACLSN (privilege class name) in an Access File, the Data Adapter for InfoMan uses it. If you do not, the Data Adapter for InfoMan inputs a PICAUSRN with a default eight-character value of SAMPID, and a PICACLSN with a default eight-character value of MASTER. During installation, these default values must be defined and recognized by the Information/Management data source. The AUTOIMAN facility automatically creates the Access File with the specified PICAUSRN and PICACLSN values.

The PICASESS (session member name) is also defined in the Access File. If you do not specify a PICASESS value in the Access File, the default session name BLGSES00 is used. In order for your Information/Management data source to recognize the default session name, you can define the default session BLGSES00 to your Information/Management data source with full MASTER class privileges. For details, see *How to Set User and Session IDs and Privilege Class Names for InfoMan* on page 3-45.

For related information, see *How to Override PICA Defaults in the InfoMan Access File* on page 3-46.

Syntax

How to Specify PICATINT in an InfoMan Access File

The PICATINT (transaction time interval) is the maximum time in seconds for a transaction to complete in the IBM API. Specify this parameter in the Access File as follows:

```
TINT=time_interval
```

where:

time_interval

Is the transaction time interval in seconds. The default value is 0.

Procedure

How to Set User and Session IDs and Privilege Class Names for InfoMan

The following steps are required to set up the default Data Adapter for InfoMan user ID (or application ID), the default privilege class name, and the default session ID.

1. Define SAMPID as an eligible user ID accessible and available to the Information/Management data source. The Data Adapter for InfoMan will interface to the low-level API with:

   ```
   PICAUSRN='SAMPID'
   ```

   You may use the Information/Management ISPF panels to do this. SAMPID must also be a member of the MASTER class with full privileges.

2. Define session BLGSES00 as a session with full privileges using the Information/Management ISPF panels. This session is a CSECT, and can be created, assembled, and link-edited by your Information/Management administrator. The session name BLGSES00 must be recognized and defined in the Information/Management data source.

3. The session member BLGSES00 must be in a load library concatenation sequence available to Information/Management. Place the assembled and link-edited BLGSES00 member in STEPLIB DD of your server JCL.

Procedure How to Override PICA Defaults in the InfoMan Access File

You can override the default PICA values by specifying them in an Access File. The following table shows the PICA parameters, their Access File keywords, and their default values.

PICA Parameter	Access File Keyword	Default Value
PICAUSRN	USRN	SAMPID
PICASESS	SESS	BLGSES00
PICACLSN	CLSN	MASTER
PICADBID	DBID	5
PICATINT	TINT	0

To change a default in an Access File, edit the text that follows its keyword in the Access File. The following is an example of an Access File with some override values:

```
DBID=5,ITABLE=TS0032I,RTABLE=TS0032R,TINT=600,
SESS=MYSESS1,USRN=JOHN,CLSN=MASTER,$
```

In this example, DBID (database ID) and CLSN (privilege class name) specify the default values. SESS (session member name) has an override value that replaces BLGSES00, USRN (user ID) has an override value that replaces SAMPID, and the maximum transaction interval is set to 600 seconds.

IGNORE Parameters in the InfoMan Access File

You can include an IGNORE= parameter in the Access File to specify the processing action for data source errors.

IGNORE=ALL	Allows all data source errors to be processed as if they were not problematic. This is not recommended.
IGNORE=NONE	Allows no data source errors to pass through. A message is issued, relevant data is logged in FSTRACE, and processing is halted.
IGNORE=TRUNC	Allows only field truncation errors to be passed through. The truncated data is returned. A message is logged in FSTRACE, and processing continues.

These errors indicate logical errors in the INFOMAN data source. For more information, see the *IBM API* Manual.

Generating the Data Adapter for Millennium

You can generate a Data Adapter for Millennium Release 3.

The Data Adapter for Millennium is included on the distribution tape. All necessary load modules were created when your site installed the server. This adapter is fully reentrant and reusable as long as you link the Millennium API modules dynamically, as described in *Linking the Millennium API Modules* on page 3-52. Prior to installing the data adapter, you can customize your configuration. See *Customizing the Data Adapter for Millennium* on page 3-47.

This documentation assumes that all of your server and data adapter libraries are catalogued under the same MVS high-level qualifier. Examples use the identifier *qualif* to refer to this high-level qualifier.

Software Requirements for the Data Adapter for Millennium

Before you install the data adapter, Millennium must be installed and fully operational. If it is not, contact your Millennium administrator.

The following two libraries are among the data sets that were unloaded from the distribution tape when the server was installed:

- EDALIB.LOAD. This library contains the server and data adapter load modules.
- EDACTL.DATA. This library contains JCL, utility programs, and data needed in the data adapter installation process.

Customizing the Data Adapter for Millennium

Prior to installing the data adapter, you can optionally customize your configuration by changing the following system defaults:

- Default lead transaction ID for the Multi:Mill feature.
- Default trace option and limits.
- API entry names.

Tip: If you are satisfied with data adapter default behavior, skip this section and go on to *Installing the Data Adapter for Millennium* on page 3-51.

The job that customizes data adapter defaults consists of three steps:

Step	Establishes:
1	API entry names (affects only *dynamic* API calls). See *Linking the Millennium API Modules* on page 3-52.
2	Default lead transaction ID.
3	Trace status and limits.

For details, see *How to Customize the Data Adapter for Millennium* on page 3-48.

Syntax

How to Customize the Data Adapter for Millennium

If you do *not* need to change any of the installation defaults, disregard the following instructions and proceed to *Installing the Data Adapter for Millennium* on page 3-51.

Edit the JCL found in member CPMILLZP of the *qualif*.EDACTL.DATA data set for your site's standards. Add a JOB card and submit the job. Sample JCL follows. The numbers on the left refer to the explanatory notes that follow the JCL.

```
   //job card goes here
   //*
   //*********************************************************************
   //* REPLACE qualif.EDALIB.LOAD WITH DSN OF YOUR LIBRARY TO BE ZAPPED
   //*
   //* THIS ZAPS FOCMIL LOAD MODULE
   //* IF YOU NORMALLY LPA THIS MODULE INTO YOUR SYSTEM YOU MUST
   //* STEPLIB TO THE ZAPPED MODULE, MLPA THE ZAPPED MODULE, OR PLPA
   //* THE ZAPPED MODULE.
   //*
   //* THIS ZAP IS SET FOR THE 3.00 VERSION OF THE MODULE
   //* IF THE VERIFIES FAIL, LOOK IN THE FIRST DUMP FOR THE VERSION
   //* OF YOUR MODULE, AND CONTACT YOUR VENDOR.
   //*********************************************************************
   //APINAMES EXEC PGM=AMASPZAP
   //SYSPRINT DD SYSOUT=*
1. //SYSLIB   DD DISP=SHR,DSN=qualif.EDALIB.LOAD
   //SYSIN    DD *
   NAME FOCMIL   FOCMIL
   *
   * VERIFY VERSION
   *
   VER 0004 1CF3,F0F0                  VERSION 3.00
   /*
2. //APINAMES EXEC PGM=AMASPZAP,COND=(0,LT)
   //SYSPRINT DD SYSOUT=*
1. //SYSLIB   DD DISP=SHR,DSN=qualif.EDALIB.LOAD
```

```
   //SYSIN    DD *
   * SET API ENTRY NAMES OR RESET VCONS
   * THERE ARE 8 SETS OF ENTRIES,
   * ZERO VCONS INDICATE THAT THE API FUNCTION IS DYNAMIC
   * BLANK ENTRY NAMES INDICATE THE FUNCTION IS NOT SUPPORTED BY THE API
   *
   NAME FOCMIL   APICVT$$
   *
   * API ENTRIES
   *
3. REP 0000 0000,0000                  INITIALIZATION VCON - DYNAMIC
   REP 0004 E7F3,E7C2,E2D9,C340        INITIALIZATION NAME - X3XBSRC
   REP 0020 0000,0000                  OPEN           VCON - DYNAMIC
   REP 0024 4040,4040,4040,4040        OPEN           NAME - NA
   REP 0040 0000,0000                  LOCATE         VCON - DYNAMIC
   REP 0044 E7F3,E7C2,C9D6,4040        LOCATE         NAME - X3XBIO
   REP 0060 0000,0000                  GENERIC GET    VCON - DYNAMIC
   REP 0064 E7F3,E7C2,C9D6,4040        GENERIC GET    NAME - X3XBIO
   REP 0080 0000,0000                  SEQUENTIAL GET VCON - DYNAMIC
   REP 0084 E7F3,E7C2,C9D6,4040        SEQUENTIAL GET NAME - X3XBIO
   REP 00A0 0000,0000                  DIRECT GET     VCON - DYNAMIC
   REP 00A4 E7F3,E7C2,C9D6,4040        DIRECT GET     NAME - X3XBIO
   REP 00C0 0000,0000                  CLOSE          VCON - DYNAMIC
   REP 00C4 E7F3,E7C2,C9D6,4040        CLOSE          NAME - X3XBIO
   REP 00E0 0000,0000                  FINISH         VCON - DYNAMIC
   REP 00E4 4040,4040,4040,4040        FINISH         NAME - NA
   /*
   //*
4. //TRANSACT EXEC PGM=AMASPZAP,COND=(0,LT)
   //SYSPRINT DD SYSOUT=*
1. //SYSLIB   DD DISP=SHR,DSN=qualif.EDALIB.LOAD
   //SYSIN    DD *
   *
   * DEFAULT LEAD TRANSACTION
   *
   NAME FOCMIL   TRANSACT
5. REP 0004 D4F3,D3D3                  LEAD TRANSACTION - M3LL
   /*
   //*
6. //TRACECTL EXEC PGM=AMASPZAP,COND=(0,LT)
   //SYSPRINT DD SYSOUT=*
1. //SYSLIB   DD DISP=SHR,DSN=qualif.EDALIB.LOAD
   //SYSIN    DD *
   *
   * TRACE CONTROL
   * THE FIRST BYTE ALLOWS TRACING IF NOT ZERO
   * THE NEXT FULL WORD SET THE LIMIT ON THE AMOUNT OF TRACING
```

```
   *
   NAME FOCMIL   TRACECTL
7. REP 0000 FF                    TRACE EVENTS - YES
8. REP 0004 FFFF,FFFF             MAXIMUM TRACE EVENTS WRITTEN - 999999999
   /*
```

where:

qualif

Is the high-level qualifier for the data sets.

1. Edit these statements to provide the high-level qualifier for your site's server production libraries.
2. This statement begins the step that establishes API entry names. It affects only dynamic API calls.
3. These REP statements specify the default API entry names. They are:

API Function	Description	Default
INITIALIZATION	File initialization	X3XBSRC
OPEN	Open for file	Not supported by the API
LOCATE	Initial position for file	X3XBIO
GENERIC GET	Perform partial key retrieval	X3XBIO
SEQUENTIAL GET	Retrieve next physical record	X3XBIO
DIRECT GET	Perform exact key retrieval	X3XBIO
CLOSE	Close for file	X3XBIO
FINISH	Perform final file closure	Not supported by the API

If you want to change these entry names, edit the REP statements.

4. This statement begins the step to establish the default lead transaction ID.
5. This REP statement makes M3LL the default lead transaction ID. If you want a different default, edit this REP statement.
6. This statement begins the step to establish trace options.
7. By default, traces are allowed. To prohibit traces, change the string 'FF' to '00'.
8. By default, unlimited trace events are allowed. To limit the amount of trace output, change the string 'FFFF,FFFF' to any hexadecimal value between '0000,0000' and 'FFFF,FFFF'.

Installing the Data Adapter for Millennium

The installation process consists of two steps. If you need a customized configuration, you must follow the instructions in *Customizing the Data Adapter for Millennium* on page 3-47 before executing the following installation steps:

1. Create the Access File library. See *Creating the Access File Library* on page 3-51.
2. Link the Millennium API modules. See *Linking the Millennium API Modules* on page 3-52.

Creating the Access File Library

At run time, the data adapter must have access to five libraries. Four of these libraries were created when the server was installed. You must now create the remaining library.

At execution time, users allocate these libraries in the server JCL, as described in *Chapter 6, Testing Server Communications*.

Each data adapter user must have access to the following libraries:

Library	Description
qualif.EDALIB.LOAD	The *qualif*.EDALIB.LOAD library was created when the server was installed. It contains the server and data adapter load modules.
MASTER	The MASTER library was created when the server was installed. At execution time, users allocate this library to ddname MASTER. Its members are Master Files. A Master File describes the fields in a Millennium data source. Each data source accessed by the data adapter must be described to the server in a Master File.
FOCEXEC	The FOCEXEC library was created when the server was installed. At execution time, users allocate this library to ddname FOCEXEC. Its members are FOCUS procedures (FOCEXECs).
ERRORS	The ERRORS library was created when the server was installed. At execution time, users allocate this library to ddname ERRORS. It contains the text of all error messages generated by FOCUS.
Access File	You must create *qualif*.FOCPRV.DATA, the Access File library. At execution time, users allocate it to ddname FOCPRV. Each member in the Access File data set corresponds to a particular member in the Master File data set. An Access File assigns the database ID and/or lead transaction ID to use with its corresponding Master File.

Example Creating the Access File Library Using the IEFBR14 Utility

The following example illustrates how to create the Access File library using the IEFBR14 utility:

```
//STEP1  EXEC PGM=IEFBR14
//FOCPRV DD   DSN=qualif.FOCPRV.DATA,DISP=(NEW,CATLG),
//            DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=1600),
//            SPACE=(TRK,(p,s,d)),VOL=SER=volser
```

where:

qualif

Is the high-level qualifier for the data sets.

p

Is the primary space allocation. Usually, 5 tracks are sufficient.

s

Is the secondary space allocation. Usually, 1 track is sufficient.

d

Is the allocation for directory blocks. Usually, 45 is sufficient.

volser

Is a valid volume ID for your site.

Many sites create private and public versions of the FOCEXEC, MASTER, and FOCPRV libraries to make maintenance easier. For example, you can create private FOCEXEC libraries for users personal sets of FOCEXECs, and retain one general site FOCEXEC library for all users to access.

If you want private and public libraries, you can create the additional libraries now. First, check to see if private libraries already exist. The additional libraries should have the same DCB attributes as the original libraries.

Linking the Millennium API Modules

This step makes the Millennium API modules available to the server through dynamic linking. Dynamic linking enables the data adapter to support all combinations of addressing options while remaining fully reentrant and reusable. For details, see *How to Link the Millennium API Modules* on page 3-53.

Syntax

How to Link the Millennium API Modules

To enable dynamic linking, edit the JCL found in member CPMILLK3 to conform to your site's standards. Add a job card and submit the job. The sample JCL follows:

```
//job card goes here
//S1       EXEC PGM=HEWLKED,PARM='LIST,LET,XREF'
//CPMILL   DD   DISP=SHR,DSN=millen.objlib
//SYSLIB   DD   DISP=SHR,DSN=millen.genlib
//SYSPRINT DD   SYSOUT=*
//SYSUT1   DD   UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSLMOD  DD   DISP=SHR,DSN=millen.loadlib
//SYSLIN   DD   *
INCLUDE CPMILL(X3XBSRC)
MODE AMODE(31),RMODE(24)
NAME X3XBSRC(R)
/*
//S2       EXEC PGM=HEWLKED,PARM='LIST,LET,XREF'
//CPMILL   DD   DISP=SHR,DSN=millen.objlib
//SYSLIB   DD   DISP=SHR,DSN=millen.genlib
//SYSPRINT DD   SYSOUT=*
//SYSUT1   DD   UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSLMOD  DD   DISP=SHR,DSN=millen.loadlib
//SYSLIN   DD   *
INCLUDE CPMILL(X3XBIO)
MODE AMODE(31),RMODE(24)
NAME X3XBIO(R)
/*
```

where:

millen

Is the high-level qualifier for your Millennium production data sets.

objlib

Is the Millennium object module library.

genlib

Is the Millennium subprogram load library.

loadlib

Is the Millennium load module library.

Reference Millennium Run-Time Requirements and Server Considerations

The data adapter now resides on your system. Before you invoke the data adapter, you must create Master Files and, optionally, Access Files for all Millennium DBID data sources that users will access, as described in the *iWay Server Administration for MVS and VM* manual. You can use the Cobol-to-FD translator to create the Master Files. See Chapter 7, *Configuring Supported Services*, for instructions on installing Cobol-to-FD translator.

You must allocate all Millennium control and data source ddnames in the server JCL, because Millennium software is unaware of any server ddname translations that are in effect.

Generating the Data Adapter for MODEL 204

Edit and run *qualif*.EDALIB.DATA(GENE204) to link-edit IFAM2 stubs into the MODEL 204 Interface. The following is the JCL to be edited:

```
//*********************************************************************
//* Name:     GENE204
//*
//* Function: Linkedit IFAM2 stubs into MODEL204 interface
//*
//* Substitutions - Change "mod204.OBJLIB" to the name of your Model204
//*                 object library.
//*               - Change "qualif" into the high-level qualifier of
//*                 your data sets.
//*********************************************************************
//*
//LKED     EXEC PGM=IEWL,PARM='LIST,NOXREF,LET'
//M204     DD   DISP=SHR,DSN=mod204.OBJLIB
//SYSLMOD  DD   DISP=SHR,DSN=qualif.EDALIB.LOAD
//MAINTAIN DD   DISP=SHR,DSN=qualif.EDALIB.DATA
//SYSUT1   DD   UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   *
  INCLUDE M204(IFIF)
  INCLUDE SYSLMOD(M204IN)
  INCLUDE MAINTAIN(M204IN)
  NAME M204IN(R)
/*
```

where:

mod204.OBJLIB

Is the name of your MODEL 204 object library.

qualif

Is the high-level qualifier for the data sets.

Generating the Data Adapter for Nomad

To generate the Data Adapter for NOMAD on MVS, perform the following steps:

1. Edit and submit *qualif*.EDALIB.DATA(NMDPAS2) to link the NOMAD library to the load library.

```
//ST008  EXEC PGM=HEWLKED,PARM='LET,XREF,LIST,MAP,SIZE=1024K'
*
//SYSUT1    DD UNIT=VIO,SPACE=(CYL,(10,1))
*
//PATCHOB1  DD DSN=qualif.EDASQL.V5R1M00.EDALIB.LOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//OLDMOD    DD DSN=qualif.EDASQL.V5R1M00.EDALIB.LOAD,DISP=SHR
//SYSLIB    DD DSN=qualif.NOMADTST.LOADLIB,DISP=SHR
//SYSLMOD   DD DSN=qualif.EDASQL.V5R1M00.EDALIB.LOAD,DISP=SHR
//SYSLIN    DD *
//   INCLUDE OLDMOD(NMDPAS2)
//   INCLUDE SYSLIB(NPI2)
//   MODE AMODE(31),RMODE(24)
//   ENTRY NPI2CALL
//   NAME NMDPAS2(R)
```

where:

qualif

Is the high-level qualifier for the data sets.

2. Edit and submit *qualif*.EDALIB.DATA(NMDIN) as follows

```
//ST007  EXEC PGM=HEWLKED,PARM='LIST,MAP,XREF,SIZE=1024K'
//SYSUT1     DD UNIT=VIO,SPACE=(CYL,(10,1))
//PATCHOBJ   DD DSN=qualif.EDASQL.V5R1M00.PTC76935.DATA,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//OLDMOD     DD DSN=qualif.EDASQL.V5R1M00.EDALIB.LOAD,DISP=SHR
//MAINTAIN   DD DSN=qualif.EDASQL.V5R1M00.EDALIB.LOAD.DATA,DISP=SHR
//SYSLMOD    DD DSN=qualif.EDASQL.V5R1M00.EDALIB.LOAD,DISP=SHR
//SYSLIN     DD *
   INCLUDE PATCHOBJ(T726C234) NMDBLD   55828  55828  C72   SAM  971020
   INCLUDE PATCHOBJ(T726C235) NMDCALL  55828  55828  C72   SAM  971020
   INCLUDE PATCHOBJ(T726C236) NMDDMP   55828  55828  C72   SAM  971020
   INCLUDE PATCHOBJ(T726C237) NMDGT    55828  55828  C72   SAM  971020
   INCLUDE PATCHOBJ(T726C238) NMDIN    55828  55828  C72   SAM  971020
   INCLUDE PATCHOBJ(T726C239) NMDOP    55828  55828  C72   SAM  971020
   INCLUDE PATCHOBJ(T726C23A) NMDSTR   55828  55828  C72   SAM  971020
   INCLUDE PATCHOBJ(T726C23B) NMDUTL   55828  55828  C72   SAM  971020
   INCLUDE OLDMOD(NMDIN)
   INCLUDE MAINTAIN(NMDIN)
    NAME NMDIN(R)
```

where:

qualif

Is the high-level qualifier for the data sets.

3. Run the following job to generate the passthru module to NOMAD for the data adapter. You must have access to the NOMAD load library before running this job. The jobs are linked into the EDALIB load library.

```
//ZAP      EXEC PGM=AMASPZAP
//SYSLIB   DD DSN=qualif.EDASQL.V5R1M00.EDALIB.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
NAME N2NPI MVSNPI2
VER 88 4770
REP 88 47F0
NAME NOMAD2 MVN2BOOT
VER 035A 4770
REP 035A 47F0
VER 0E70 4770
REP 0E70 47F0
*NAME MVN2BOOT ALIAS TO NOMAD2
*VER 035A 4770
*REP 035A 47F0
*VER 0E70 4770
*REP 0E70 47F0
NAME NOMSYS $FUNC
VER 0470 4770
REP 0470 47F0
//
```

where:

qualif

Is the high-level qualifier for the data sets.

Generating the Data Adapter for Oracle

Edit and run *qualif*.EDALIB.DATA(GENEORA) to link-edit the Data Adapter for Oracle . The following is the JCL to be edited:

```
//*********************************************************************
//* Name:     GENEORA
//*
//* Function: Linkedit the server ORACLE interface
//*
//* Substitutions: - Change "qualif" into the high-level qualifier
//*                  for your data sets.
//*                - Change "oraqual.SQLLIB" to the name of your
//*                  Oracle load library.
//*********************************************************************
//ASM      EXEC PGM=IEV90,PARM=(OBJECT,NODECK,NOLIST)
//SYSLIB   DD   DISP=SHR,DSN=SYS1.MACLIB
//SYSLIN   DD   DISP=(MOD,PASS),DSN=&&ORAPRM,UNIT=SYSDA,
//         SPACE=(100,(50,50)),DCB=(LRECL=80,BLKSIZE=800)
//*SYSGO   DD   DDNAME=SYSLIN
//SYSUT1   DD   UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DISP=SHR,DSN=qualif.EDALIB.DATA(ORAPARMS)
//*
//LKED     EXEC PGM=IEWL,PARM='LIST,NOXREF,LET'
//SYSLIB   DD   DISP=SHR,DSN=oraqual.SQLLIB
//OBJECT   DD   DISP=SHR,DSN=qualif.EDALIB.DATA
//SYSLMOD  DD   DISP=SHR,DSN=qualif.EDALIB.LOAD
//PARMS    DD   DISP=(OLD,DELETE),DSN=&&ORAPRM,UNIT=SYSDA
//SYSUT1   DD   UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   *
  INCLUDE OBJECT(ORAPAS2)
  INCLUDE SYSLIB(HLISTUBC)
  ENTRY ORAPAS2
  NAME ORAPAS2(R)
  REPLACE JJCOM2@
  REPLACE JJCOM2
  INCLUDE SYSLMOD(ORAFOC)
  INCLUDE PARMS
  INCLUDE OBJECT(ORAFOC)
  NAME ORAFOC(R)
/*
```

where:

oraqual.SQLLIB

Is the name of your Oracle load library.

qualif

Is the high-level qualifier for the data sets.

Generating the Data Adapter for SUPRA

The SUPRA read-only data adapter supports SUPRA Release 2.1 and higher.

For users of Release 1.35, rename *qualif*.EDALIB.LOAD(CSTEDCMX) to *qualif*.EDALIB.LOAD(CSTEDCMT).

Link-edit the supplied Data Adapter for SUPRA module with the SUPRA stub called CSVILUV. The CSVILUV module is supplied by Cincom and usually resides in the SUPRA LINKLIB library. This procedure is necessary and allows the Data Adapter for SUPRA module to communicate with the SUPRA Central PDM. The following link-edit JCL should be modified to conform to site standards and submitted for execution. The JCL listed below can be found in the *qualif*.EDALIB.DATA(GENEFSP) data set.

```
//*********************************************************************
//* Name:     GENEFSP
//*
//* Function: Linkedit Supra stubs into Supra interface
//*
//* Substitutions: - Change "supra.LINKLIB" to the name of your Supra
//*                  object library.
//*                - Change "qualif" to the high-level qualifier of
//*                  your libraries.
//*********************************************************************
//*
//LKED     EXEC PGM=IEWL,PARM='LIST,NOXREF,LET'
//SUPRA    DISP=SHR,DSN=supra.LINKLIB
//SYSLMOD  DISP=SHR,DSN=qualif.EDALIB.LOAD
//MAINTAIN DISP=SHR,DSN=qualif.EDALIB.DATA
//SYSUT1   DD  UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  *
//SYSLIB   DD  DUMMY
  INCLUDE SUPRA(CSVILUV)
  INCLUDE SYSLMOD(FSP000)
  INCLUDE MAINTAIN(FSP000)
  NAME FSP000(R)
/*
```

where:

supra.LINKLIB

Is the name of the SUPRA load library that contains the CSVILUV module.

qualif

Is the high-level qualifier for the data sets.

Configuring the SUPRA Multi-Session Option

The SUPRA read-only data adapter supports SUPRA Release 2.1 and higher.

For users of Release 1.35, rename *qualif*.EDALIB.LOAD(CSTEDCMX) to *qualif*.EDALIB.LOAD(CSTEDCMT).

No security is provided by the multi-session interface. However, the multi-session facility does provide a security exit that can be used to control access by users to the data source resource. This security exit is identified with the SECURITY= keyword. See *SUPRA Multi-Session Security Block* on page 3-61.

You must edit the member SUPRCNFG, which resides in the *qualif*.EDALIB.DATA data set. This member contains the parameters to be used in the connection of the multi-session interface to the Central PDM. All keywords must start in column 1 and each must be specified on a separate input statement. An asterisk (*) in column 1 indicates a commented line. For a list of valid options, see *SUPRA Multi-Session Keywords* on page 3-59.

Reference SUPRA Multi-Session Keywords

Listed below are valid options to use in editing the member SUPRCNFG, where *xx* are values to be supplied by the user.

Keyword	Description
`THREADS=xx`	Is the number of concurrent requests that can be issued to the PDM. If this number is too small, a TFUL status may result. See the *SUPRA PDM Administrator Guide* for more information.
`TASK=xx`	Is the maximum number of tasks that can be signed on to the PDM at any one time, including two tasks: one for the Autostart facility and one for data adapter tracking. Therefore, if TASK=10, up to eight users can be signed on. If this value is exceeded, a CFUL or MFUL status may result. The MFUL status is from the multi-session interface and indicates that the task tables are full. A CFUL is from the PDM and indicates the same for the PDM.
`SECURITY=xxxxxxx`	Is the name of the user-written security module that will validate access of users to the PDM data resources. If provided, it must reside in a library accessible from the multi-session interface. For related information, see *SUPRA Multi-Session Security Block* on page 3-61.
`DEBUG=xxx`	Is a diagnostic facility that will provide console messages for problem tracing. See the section *DEBUG Parameter* in your *SUPRA Usage Instructions* for details.

Keyword	Description
`END.`	Is the ending statement for the parameters. The period is required.

The *qualif*.EDALIB.LOAD library contains several load modules that make up the multi-session interface. These modules are as follows:

Module	Description
`CFDP4001`	This is a new load module developed for support of the multi-session interface. This module is link-edited with the name CSTEDBMI. This is the same name as the current Cincom single task and central region Data Adapters. It will be necessary to place *qualif*.EDALIB.LOAD, the data set that contains this module, ahead of the LINKLIB and INTERFLM load libraries supplied by Cincom. This module is self-contained; it is not a composite as are the other CSTEDBMI modules. Care should be exercised not to overlay one of the other Cincom Data Adapters.
`CFDP4002`	This is the reusable load module that provides the link to the multi-session interface. It must be linked as reusable so that only one copy is in the address space. It is loaded by the CFDP4001 module when the first PDM request is processed. This module loads the multi-session interface module (CSTEDBMI) and establishes the connection with the Central PDM. It reads the input parameters provided to determine the number of tasks and concurrent operations that will be supported during the session. Any errors that are found will generate a console message. Depending on the error, the connection may be terminated.

Module	Description
CFDP4003	This is the security module that provides the exit for the customer to control access to the PDM data resources. It is loaded by module CFDP4002 when the input parameters specify the SECURITY= keyword. It loads the named user module and passes a security block to the module. The user module is called using the standard IBM calling conventions with register 1 pointing to the parameter list and register 13 pointing to the register save area. The passed area is described in the next table. On return from the called security module, the status field is examined. The operation is accepted if the status field is ****. Any other status will result in the operation being terminated with a status of SECR. The user security module may not issue any calls to the PDM. **Note:** (**** = ACCESS ALLOWED, ANYTHING ELSE = NOACCESS)

Reference

SUPRA Multi-Session Security Block

The Supra multi-session facility provides a security exit that can be used to control access by users to the data source resource.

Field Name	Field Format	Field Length	Starting Position
USER-ID	Character	8	1
ACCESS	Character (R=Read, W=Write)	1	9
DD-NAME	Character	4	10
DSN-NAME	Character	44	14
STATUS	Character	4	58

Configuring the SUPRA Multi-Session Batch Autostart Facility

When the multi-session option for SUPRA is used with the server, the first task that issues a request to the Central PDM will be used by the multi-session interface to establish a permanent connection between the Central PDM and the server region. The initial request, if executed from a client logged on to the server, would tie up that client until the region was shut down. To avoid this scenario, the Batch Autostart Facility should be used to initialize and establish the connection between the server and Central PDM regions.

Note: If you bring up the server before the SUPRA PDM is up, the Autostart connection will not be established. Also, if the PDM is recycled, the Autostart connection will be broken. To connect under these circumstances, use the Console to restart the service block associated with the Autostart task (SERVICE=GATEKEPR).

The server configuration file that is allocated to the ddname EDASERVE must be modified to include an additional service block. This service block will be used by the server to execute an "exec" at startup, which will run as a background task and establish the link between the server and the PDM. The syntax for adding the additional service block is shown below:

```
SERVICE                = GATEKEPR
PROGRAM                = TSCOM3
NUMBER_READY           = 1
MAXIMUM                = 1
RUNCOUNT               = 1
SERVINIT               = *,++
  AUTOSTART            = BATCH
  DYNAM ALLOC FILE STDOUT SYSOUT X LRECL 132 BLKSIZE 132 RECFM F
  DYNAM ALLOC DD SYSPRINT SYSOUT X REU CLOSE
  DYNAM ALLOC FILE TRMIN DA qualif.EDARPC.DATA(member) SHR REU
  DYNAM ALLOC FILE TRMOUT SYSOUT X LRECL 133 BLKSIZE 133 RECFM F
++
```

where:

`SERVICE`

Identifies and names a service. The name you choose should be unique and from 1 to 8 characters in length. The keywords and values that follow the SERVICE keyword in each service block define the type of service to be provided.

`PROGRAM`

This parameter should always be set to TSCOM3.

`NUMBER_READY`

Is the number of instances of this service to be preloaded by the server. This parameter should always be set to 1.

`MAXIMUM`

Is the number of instances of this service allowed by the server. This parameter should always be set to 1.

`RUNCOUNT`

Specifies the number of attempts to connect to the central PDM. There is no default value. Therefore, you must specify a value to prevent the server from continually attempting to access SUPRA data.

`SERVINIT`

Indicates the "in stream" service initialization file to invoke for this service.

`qualif`

Is the high-level qualifier for the data sets.

`member`

This member should contain the following:

```
SQL
SELECT BRANCH_NAME FROM BRANCH;
END
```

This request will not retrieve any data but is used to maintain the thread to SUPRA.

Reference Supplementary Code for Data Adapter for SUPRA JCL

The following supplements the JCL that you will edit in *Chapter 6, Testing Server Communications*.

```
//STEPLIB  DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//         DD DSN=supra.INTERFLM,DISP=SHR
//         DD DSN=supra.LINKLIB,DISP=SHR

//CSIPARM  DD DSN=supra.SAMPLE.CSIPARM(TXPFOCUS),DISP=SHR
//CSISYSIN DD DSN=qualif.EDALIB.DATA(SUPRCNFG),DISP=SHR
```

You must also allocate all SUPRA Master and Access File definitions to ddnames MASTER and ACCESS, respectively.

Generating the Data Adapter for SYSTEM 2000

Edit and run *qualif*.EDALIB.DATA(GENES2K) to re-link S2KPL into S2K, which will be called by the server. The following is the JCL to be edited:

```
//*********************************************************************
//*Name:     GENES2K
//*
//* Function: Linkedit System2000 stubs into the S2K interface
//*
//* Substitutions: - Change "system2000.LOAD" to the name of your
//*                  System2000 load library.
//*                - Change "qualif" to the high-level qualifier
//*                  for your data sets.
//*********************************************************************//*
//LKED      EXEC PGM=IEWL,PARM='LIST,NOXREF,LET,MAP'
//S2K       DD   DISP=SHR,DSN=system2000.LOAD
//SYSLMOD   DD   DISP=SHR,DSN=qualif.EDALIB.LOAD
//MAINTAIN  DD   DISP=SHR,DSN=qualif.EDALIB.DATA
//SYSUT1    DD   UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT  DD   SYSOUT=*
//SYSLIN    DD   *
INCLUDE S2K(S2KPL)
INCLUDE SYSLMOD(S2K)
INCLUDE MAINTAIN(S2K)
NAME S2K(R)
```

where:

system2000.LOAD

Is the name of the SYSTEM 2000 load library.

qualif

Is the high-level qualifier for the data sets.

Generating the Data Adapter for Teradata

Edit and run qualif.EDALIB.DATA(GENEDBC). The following is the JCL to be edited:

```
//*********************************************************************
//* Name:     GENEDBC
//*
//* Function: Assemble and linkedit the Teradata interface
//*
//* Substitutions: - Change "qualif" into the high-level qualifier
//*                  for your data sets.
//*                - Change "teradata.APPLOAD" to the name of your
//*                  Teradata load library.
//*                - Change the "IM" parameter to indicate the interface
//*                  support for user-issued native SQL commands:
//*                          '0' (support disabled) or
//*                          '2' (support fully enabled).
//*                - Change the "REL" parameter to indicate the Teradata
//*                  software release:
//*                          '4' (any rel. 4 level) or
//*                          '3' (any rel. 3 level).
//*                 Note that this parameter must be exactly 4 bytes long
//*********************************************************************
//DBCPROC PROC PREFIX='qualif',
//             IM='2'
//             REL='220 '
//*
//ASM    EXEC PGM=IEV90,PARM=(OBJECT,NODECK,NOLIST,'SYSPARM(&IM&REL)')
//SYSLIB   DD   DISP=SHR,DSN=SYS1.MACLIB
//SYSIN    DD   DISP=SHR,DSN=&PREFIX..EDALIB.DATA(DBTPARMS)
//SYSLIN   DD   DISP=(MOD,PASS),DSN=&&DBCPRM,UNIT=SYSDA,
//         SPACE=(100,(50,50)),DCB=(LRECL=80,BLKSIZE=800)
//*SYSGO   DD   DDNAME=SYSLIN
//SYSUT1   DD   UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSPRINT DD   SYSOUT=*
//*
//LKED     EXEC PGM=IEWL,PARM='LIST,NOXREF,LET',COND=(4,LT,ASM)
//SYSLIB   DD   DISP=SHR,DSN=teradata.APPLOAD
//MAINTAIN DD   DISP=SHR,DSN=&PREFIX..EDALIB.DATA
//SYSLIN   DD   DISP=SHR,DSN=&PREFIX..EDALIB.DATA(CLIV2)
//PARMS    DD   DISP=(OLD,DELETE),DSN=&&DBCPRM,UNIT=SYSDA
//SYSLMOD  DD   DISP=SHR,DSN=&PREFIX..EDALIB.LOAD
//SYSUT1   DD   UNIT=SYSDA,SPACE=(100,(50,50))
//SYSPRINT DD   SYSOUT=*
//*
//DBCPROC   PEND
//GENFDBC   EXEC DBCPROC
```

where:

qualif

Is the high-level qualifier for the data sets.

IM

Indicates the interface support for user-issued native SQL commands. Possible values are:

'0' indicates support is disabled.

'2' indicates support is fully enabled.

REL

Indicates the Teradata software release level. This parameter must be exactly 4 bytes long.

'4' represents any release 4 level or above.

'3' represents any release 3 level.

teradata.APPLOAD

Is the name of your Teradata load library.

CHAPTER 4

Generating a Server

Topics:

- Generating a Server Type
- Generating a Hub Server
- Generating a Full-Function Server
- Next Steps

The Server for MVS provides procedures that you can use to create the server configuration files. The procedures include editing instructions and descriptions of how the resulting configuration files interact.

This information will be useful if you decide to customize your configuration further, as described in Chapter 8, *Modifying the Server Configuration*.

Generating a Server Type

The *qualif*.EDALIB.DATA data set contains configuration procedures for generating each of the possible server types. The following members of *qualif*.EDALIB.DATA generate the configuration files for a corresponding server type:

Server Type	Member Name	Page #
Hub Server	EDACFGH	*Generating a Hub Server* on page 4-4
Full-Function Server	EDACFGF	*Generating a Full-Function Server* on page 4-12

These procedures generate the startup configuration file (EDASERVE), the communications file (EDACSG) and the server JCL for starting the server. They also create jobs for testing your configuration. See Chapter 6, *Testing Server Communications*, for more information about these jobs.

After the procedure is complete, check the job output for SYSTSPRT ddnames or a return code of 99 (both indicate that an error has been detected). If one or more errors exist, check the SYSTSPRT file and correct the error. The following errors, and most other problems, are usually associated with edit errors in the procedure. These procedures produce the following error messages

```
EDARSERV *** ERROR *** License Key not 12 chars long : - license

EDARSERV *** ERROR *** Owner value must be 1-7 long  : - owner

EDARSERV *** ERROR *** Server type invalid          : - servertype
EDARSERV *** ERROR *** Must be RDB2, RDBC, HUB or FFS

EDARIMS *** ERROR *** JCL file line not found
EDARIMS *** ERROR *** Cannot edit for IMS DBCTL activation
```

where:

license

Is the License Key that you supplied in EDAQSERV. It is the 12-digit, three-part license number included on an insert in the software package. If you receive this error, you must correct the EDACFGx procedure and resubmit the job.

owner

Is the owner_id of the catalog tables that you supplied in EDAQSERV. It must be 1-7 characters in length. If you receive this error, you must correct the EDACFGx procedure and resubmit the job.

`servertype`

Is an incorrect server type value. Valid server type values are RDB2, RDBC, HUB, or FFS. If you receive this error, you must correct the EDACFGx procedure and resubmit the job.

Note: If EDARSERV (a REXX routine called by the EDAQSERV procedure) ends due to an error, the additional output members of *qualif*.INSTALL.DATA will be generated with one record indicating that EDAQSERV ended with errors.

The EDACFGx procedures allow you to configure inbound and outbound node blocks for the applicable server types. A sample procedure for each valid protocol is included in the EDACFGx procedure. For inbound communications, you can have only one block per protocol. For outbound communications, you can have multiple blocks for the same protocol. To create multiple outbound blocks for a protocol, copy the procedure for that protocol and add it to the EDACFGx procedure. For example, if you want to configure multiple outbound TCP/IP blocks, copy the EDAQTCPO procedure as many times as necessary and edit the blocks for the servers to which you want to connect.

Once you have a working server, you should manually add any new communications blocks to the EDACSG file instead of re-running the EDACFGx procedure. If you do run the EDACFGx procedure again, in order to add a communications node block, you erase all customization of the current server configuration. See Chapter 8, *Modifying the Server Configuration*, for information on adding communications node blocks.

The EDACFGx procedures create output members in *qualif*.INSTALL.DATA. The only exceptions to this are the server profiles, which are created in *qualif*.EDAPROF.DATA. Refer to your worksheets during the configuration process.

Specifying Values for an Ownerid

In the configuration procedures EDACFGF and EDACFGH, an *ownerid* is required in several places. This *ownerid* is used as part of the Dynamic Catalog data set names and profile member names. The value specified for *ownerid* must be the same in all the parameters that require it. If a different value is used, for example, in EDAQCFGF and EDAQSERV procedures, an allocation reference to a nonexistent member could result. This causes an S013 abend when a user attempts to connect to the server.

The following message may appear in the JES2 log for any or all of the EDACFGx configuration routines:

```
+TCB:*JSTCB R1FNS  :(F0C35802) FAILURE TO READ EDA SERVER LIST CORRECTLY
```

This message is informational only and can be ignored.

Generating a Hub Server

To generate the configuration files for a Hub Server, you must edit and submit *qualif*.EDALIB.DATA(EDACFGH). This job executes the following procedures:

Procedure	**Member Name in** *qualif*.**INSTALL.DATA**	**Description**
HUBDEL		Deletes data sets created by the next step for use when re-running the job. Step is commented out as supplied.
EDAQCFGH		Creates the Hub catalog used by the adapters. It creates four data sets and loads them with initial data.
EDAQSERV	HUBSERV	EDASERVE Server configuration file with one service block for general users (service name of EDAUSER) and one service block for the EDADBA (service name of EDADBA), which is used to create Master and Access Files. See the *iWay Server Administration for MVS and VM* manual for more information.
	HUBCSG	EDACSG Server communications file.
	HUBJCL	Server JCL.
	CS3CLNT	Test client for running the Installation Verification Procedure.
	RDAAPPC	CLIST test client for running RDAAPP diagnostic tests.
	RDAAPPJ	JCL test client for running RDAAPP diagnostic tests.
	JRUNIVP	Creates the JCL for the Installation Verification Program.
EDAQCON	HUBCON	Uncomment this procedure to create the Console communications file and include the user ID, specified in the OWNER parameter, as the console administrator in HUBSERV.

Procedure	Member Name in *qualif*.**INSTALL.DATA**	Description
EDAQTCPI		Uncomment this procedure to create a TCP/IP inbound communications block in the HUBCSG file.
EDAQLU6I		Uncomment this procedure to create an LU6.2 inbound communications block in the HUBCSG file.
EDAQTCPO		Uncomment this procedure to create a TCP/IP outbound node block in the HUBCSG file.
EDAQLU6O		Uncomment this procedure to create an LU6.2 outbound node block in the HUBCSG file.

Note: CS3CLNT, RDAAPPC, RDAAPPJ, and JRUNIVP are client applications. Each time you run an EDACFGx procedure, the EDACFGx procedure recreates these members. Therefore, they are valid for only the most recent EDACFGx procedure.

Syntax

How to Generate Hub Server Configuration Files

Important: You must uncomment and supply a value for all of the parameters when you are editing any of the EDAQxxxx procedures. Never leave a parameter blank (=,).

Edit *qualif*.EDALIB.DATA(EDACFGH) as follows and submit the job.

Note: All values to the right of the equal sign (=) should be in uppercase.

```
File: EDACFGH   JCL

//*        JOB CARD GOES HERE
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*Purpose: To create the configuration files and test procedures   *
//*         for a HUB Server.                                        *
//*                                                                  *
//*     HUBDEL   Deletes data sets created by the next step for use *
//*              when re-running the job.  Step is commented out    *
//*              as supplied.                                        *
//*                                                                  *
//*     EDAQCFGH Creates the HUB catalogue used by Server.          *
//*              Creates four data sets and loads them with initial *
//*              data.                                               *
//*                                                                  *
```

```
//*     EDAQSERV Creates the following Server files in              *
//*              in qualif.INSTALL.DATA                             *
//*                                                                 *
//*              HUBSERVE - Server configuration file               *
//*              HUBCSG   - Server communications file              *
//*              HUBJCL   - Server JCL                              *
//*              CS3CLNT  - RDAAPP client communications file       *
//*              RDAAPPC  - CLIST to run RDAAPP client test program *
//*              RDAAPPJ  - JCL   to run RDAAPP client test program *
//*              JRUNIVP  - JCL   to run IVP test program           *
//*              TEMP     - Temporary member used to pass values    *
//*                         between procedures                      *
//*                                                                 *
//** WARNING **  The client members above are re-created each       *
//*              time you run an EDACFGx procedure, so they will    *
//*              only be valid for the last procedure run.          *
//*                                                                 *
//*     EDAQCON  Creates the following Server file  in              *
//*              in qualif.INSTALL.DATA. Also modifies HUBSERV      *
//*              file to include single user (ownerid) of console   *
//*                                                                 *
//*              HUBCON  - Console communications file              *
//*                                                                 *
//*     EDAQTCPI Creates a TCP/IP inbound communications block      *
//*     EDAQLU6I Creates a Lu62 inbound communications block        *
//*              in qualif.INSTALL.DATA(FFSCSG). Uncomment one or   *
//*              more of these procedures depending on the          *
//*              protocol(s) you want the server to listen on       *
//*                                                                 *
//*     The following steps are used to configure the outbound      *
//*     blocks which point to the subservers that this HUB has      *
//*     access to                                                   *
//*                                                                 *
//*     Each step can be duplicated and different values specified  *
//*     so that multiple outbound node blocks for the same protocol *
//*     can be generated in one run of EDACFGH                      *
//*                                                                 *
//*     Make sure that each "NODENAME" specified is UNIQUE          *
//*                                                                 *
//*     EDAQTCPO Creates a TCP/IP outbound node block               *
//*                                                                 *
//*     EDAQLU6O Creates a LU6.2  outbound node block               *
//*                                                                 *
//*                                                                 *
//*Substitutions:                                                   *
//*                                                                 *
//*      qualif  The high level qualifier for the Server data sets. *
//*                                                                 *
```

```
//*      OWNER   Owner id of the catalogue tables.  May be from     *
//*              one to seven characters long.                      *
//*                                                                 *
//*      DUNIT   Specifies the disk unit to use when allocating the *
//*              data sets.  It is defaulted to SYSDA in this JCL.  *
//*              Should you wish to specify a volume as well as a   *
//*              unit when allocating the data sets, please use the *
//*              following form:                                    *
//*                    DUNIT='myunit,VOL=SER=myvol'                 *
//*              e.g., DUNIT='SYSDA,VOL=SER=USERM1'                 *
//*                                                                 *
//*      SERVTYPE  DO NOT CHANGE THIS VALUE                         *
//*                                                                 *
//*      LICENSEK  The license key that was supplied with the       *
//*                product                                          *
//*                                                                 *
//*      MAXIMUM   The maximum number of users for this HUB         *
//*                Server - must not exceed License agreement       *
//*                                                                 *
//*      DEPLOY    The deployment type for this server              *
//*                Values can be                                    *
//*                PRIVATE (default) Each user of the server will   *
//*                                  get a private task which will  *
//*                                  be cleared when they           *
//*                                  disconnect so the task will be *
//*                                  re-used by the next user.      *
//*                                  NOTE: PDS indexes will not be  *
//*                                  re-read in this mode. This is  *
//*                                  not true for server profiles,  *
//*                                  which are always refreshed for *
//*                                  each user connection.          *
//*                PRIVATE,REREAD      Same as for PRIVATE but all  *
//*                                  PDS files will be re-read for  *
//*                                  each user. Should only be used *
//*                                  for EDADBA service or when in  *
//*                                  development mode.              *
//*                ISOLATED          Each user of the server will   *
//*                                  get a private task which will  *
//*                                  be deleted when they           *
//*                                  disconnect from the server.    *
//*                POOLED            Each user of the server will   *
//*                                  re-use a task and context      *
//*                                  of last user of the task. Task *
//*                                  will NOT be deleted on user    *
//*                                  disconnect.                    *
//*                                                                 *
//*      SYSOUT    SYSOUT class server will use for trace file      *
//*                (When activated)                                 *
```

```
//*                                                                    *
//*      CONAPPLD  The VTAM resource name (APPLID) to be used for      *
//*                access to the EDA console                           *
//*                                                                    *
//*      TCPNAME   The TCP/IP job name. The default is TCPIP for IBM*
//*                For INTERLINK, it the subsystem name.               *
//*                                                                    *
//*      VENDOR    The name of the TCP/IP vendor. Default value is     *
//*                IBM or it can have a value of INTERLINK or OCS      *
//*                                                                    *
//*      SERVICE   The TCP/IP port number the server will listen on *
//*                                                                    *
//*      HOST      The IP address of the machine that the server       *
//*                is running on.                                      *
//*                                                                    *
//*      CLIENTLU  The VTAM resource name (APPLID) that the test       *
//*                Client will use.                                    *
//*                                                                    *
//*      SERVERLU  The VTAM resource name (APPLID) that the server     *
//*                will listen on.                                     *
//*                                                                    *
//*      MODENAME  The MODENAME that the LU6.2 APPLID was generated *
//*                with.                                               *
//*                                                                    *
//*      NODENAME  This must match the subservers service name if      *
//*                the subserver is running on MVS. For UNIX and NT *
//*                use a 1-8 character descriptive name.               *
//*                                                                    *
//*      SUBHOST   The IP address of the machine that the subserver *
//*                is running on.                                      *
//*                                                                    *
//*      SUBPORT   The port number the subserver is using to           *
//*                listen on.                                          *
//*                                                                    *
//*      LOCALLU   An APPLID used by the server so that it can act     *
//*                as a Client. You can specify '(ddname)' E.G         *
//*                (LU6POOL) which is a ddname pointing to a list      *
//*                of APPLIDs. If you used this option, you will       *
//*                need to ADD a ddname card to the generated server*
//*                JCL that points to the list of APPLIDs              *
//*                                                                    *
//*      PARTLU    The APPLID that the subserver is using to listen *
//*                on.                                                 *
//*                                                                    *
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*
//EDAPROCS JCLLIB ORDER=qualif.EDALIB.DATA
//*HUBDEL   EXEC PROC=HUBQDEL,
```

```
//*          OWNER=owner_id
//EDAQCFGH EXEC PROC=EDAQCFGH,
//         OWNER=owner_id,
//         DUNIT=SYSDA
//EDAQSERV EXEC PROC=EDAQSERV,
//         SERVTYPE=HUB,
//         LICENSEK=license key,
//         MAXIMUM=maximum users,
//         DEPLOY=PRIVATE,
//         SYSOUT=sysout class,
//         OWNER=OWNER_ID
//*EDAQCON  EXEC PROC=EDAQCON,
//*          SERVTYPE=HUB,
//*          CONAPPLD=Console applid
//*EDAQTCPI EXEC PROC=EDAQTCPI,
//*          SERVTYPE=HUB,
//*          TCPNAME=TCPIP,
//*          VENDOR=IBM,
//*          SERVICE=tcp port number,
//*          HOST=host name
//*EDAQLU6I EXEC PROC=EDAQLU6I,
//*          SERVTYPE=HUB,
//*          CLIENTLU=local lu name for client,
//*          SERVERLU=server lu name
//*          MODENAME=mode name for APPLID
//*EDAQTCPO EXEC PROC=EDAQTCPO,
//*          SERVTYPE=HUB,
//*          NODENAME=node name of subserver,
//*          TCPNAME=TCPIP,
//*          VENDOR=IBM,
//*          SUBHOST=ip address,
//*          SUBPORT=port number
//*EDAQLU6O EXEC PROC=EDAQLU6O,
//*          SERVTYPE=HUB,
//*          NODENAME=node name of subserver,
//*          LOCALLU=local lu name,
//*          PARTLU=partner lu name,
//*          MODENAME=Vtam mode name
```

where:

qualif

Is the high-level qualifier for the data sets.

owner_id

Is the owner ID of the catalog tables. It is usually the TSO Userid of the person configuring the server. For related information, see *Specifying Values for an Ownerid* on page 4-3.

`DUNIT`

Specifies the disk unit to use when allocating the data sets. The default in this JCL is SYSDA. If you want to specify a volume as well as a unit when allocating the data sets, use the following format:

```
DUNIT='myunit,VOL=SER=myvol'
```

For example,

```
DUNIT='SYSDA,VOL=SER=USERM1'
```

`HUB`

Is the Hub Server setting. Do not change this value.

`license key`

Is the license key supplied with the product.

`maximum users`

Is the maximum number of users for this Hub Server.

`deployment mode`

The deployment type for this server. The modes of deployment are:

`PRIVATE`: (default) Each user of the server receives a private task, which will be cleared when they disconnect so that the task can be reused by the next user.

Note: PDS indexes will not be reread in this mode. This is not true for server profiles, which are always refreshed for each user connection.

`PRIVATE,REREAD`: Same as for PRIVATE, but all PDS files will be reread for each user. Should be used only for EDADBA service or when in development mode.

`ISOLATED:` Each user of the server will get a private task, which will be deleted when they disconnect from the server.

`POOLED`: Each user of the server reuses the task and context of last user of the task. The server does not delete tasks when the user disconnects.

`sysout class`

Is the SYSOUT class to which the server sends trace output when tracing is activated. For example, the specification SYSOUT X sends the output to SYSOUT CLASS X.

`console applid`

Is the VTAM resource name (APPLID) used for Console access. This is an LU2 APPLID.

`tcp name`

For IBM, TCP/IP is the name of the started task or the JOB name for the TCP/IP subsystem address space, if it is different from the default (TCPIP).

For Interlink, TCP/IP is the name of the TCP/IP subsystem (no default value).

For OCS use the default value of TCPIP.

Note: The server can support only one TCP/IP vendor at one time for both inbound and outbound blocks. Use multiple servers to receive support from multiple vendors.

`vendor name`

Is the TCP/IP vendor name. Possible values are IBM, INTERLINK, or OCS.

`tcp port number`

Is the TCP/IP port number on which the server listens.

`host name`

Is the IP address of the machine on which the server runs.

`client lu name`

Is the VTAM resource name (APPLID) that the LU6.2 test client uses.

The APPLID for LU62 test clients must be unique, and cannot be the same as those specified for LU62 inbound communications (server lu name). You have the option of using a pooled list of APPLIDs by setting this value to (LU6POOL); for example, CLIENTLU=(LU6POOL). See *DDNAME LUPOOL* on page 4-12 and Chapter 6, *Testing Server Communications*, for more information.

`server lu name`

Is the VTAM resource name (APPLID) on which the server listens for LU6.2 inbound communications. This APPLID must be unique.

`mode name for APPLID`

Is the MODENAME with which the server's LU6.2 APPLID was defined.

`node name of subserver`

The name of the service on the subserver to which this server connects. If the subserver is on MVS, this value must match the name of a service specified in the subserver's EDASERVE file. You need to specify a node name for each outbound communications block. This name must be unique for each block. If the subserver is on UNIX or NT, this value should be a 1-8 character descriptive name. It is used for reference by the CREATE SYNONYM command.

`ip address`

Is the IP address of the machine on which the subserver is running.

`port number`

Is the port name or number on which the subserver is listening.

`local lu name`

Is the APPLID that the Hub Server will use to act as a client to communicate with a subserver.

If you need several client connections to a subserver, you must use DDNAME LUPOOL syntax. See *DDNAME LUPOOL* on page 4-12 and Chapter 6, *Testing Server Communications*, for more information.

`partner lu name`

Is the LU6.2 APPLID on which the subserver listens for inbound communications.

`Vtam mode name`

Is the MODENAME with which the subserver's LU6.2 APPLID was defined.

DDNAME LUPOOL

LUPOOL is a ddname that points to a list of APPLIDs. This option, available with all server types, allows a client or test client to connect using the first available APPLID from the list instead of waiting for a specific APPLID to be available. If you use this option, you need to allocate a ddname in the startup JCL, for example LU6POOL, to point to a list of APPLIDs. See Chapter 6, *Testing Server Communications*, for more information.

Generating a Full-Function Server

To generate the configuration files for a Full-Function Server, you must edit and submit *qualif*.EDALIB.DATA(EDACFGF). This job executes the following procedures:

Procedure	**Member Name in** *qualif*.**INSTALL.DATA**	**Description**
FFSDEL		Deletes data sets created by the next step for use when re-running the job. Step is commented out as supplied.
EDAQCFGF		Creates the FFS catalog used by the server. It creates four data sets and loads them with initial data.

Procedure	**Member Name in** *qualif.***INSTALL.DATA**	**Description**
EDAQSERV	FFSSERV	EDASERVE Server configuration file with one service block for general users (service name of EDAUSER) and one service block for the EDADBA (service name of EDADBA), which is used to create Master and Access Files. See the *iWay Server Administration for MVS and VM* manual for more information.
	FFSCSG	EDACSG Server communications file.
	FFSJCL	Server JCL.
	CS3CLNT	Test client for running the Installation Verification Procedure.
	RDAAPPC	CLIST test client for running RDAAPP diagnostic tests.
	RDAAPPJ	JCL test client for running RDAAPP diagnostic tests.
	JRUNIVP	Creates the JCL for the Installation Verification Program.
	TEMP	Temporary member used to pass values between procedures.
EDAQCON	FFSCON	Uncomment this procedure to create the Console communications file and include the user ID, specified in the OWNER parameter, as the console administrator in FFSSERV.
EDAQIMS		Activates server access to IMS DBCTL by modifying FFSSERV and FFSJCL. To complete DBCTL activation, you then need to edit the FOCPSB DD statement in the FFSJCL.
EDAQTCPI		Uncomment this procedure to create a TCP/IP inbound communications block in the FFSCSG file.

Procedure	Member Name in *qualif*.**INSTALL.DATA**	Description
EDAQLU6I		Uncomment this procedure to create an LU6.2 inbound communications block in the FFSCSG file.

Full-Function Server With Hub Services Only		
Procedure	**Member Name in** *qualif*.**INSTALL.DATA**	**Description**
EDAQTCPO		Uncomment this procedure to create a TCP/IP outbound node block in the FFSCSG file. This step is only required if you are configuring a Full-Function Server with Hub Services.
EDAQLU6O		Uncomment this procedure to create an LU6.2 outbound node block in the FFSCSG file. This step is only required if you are configuring a Full-Function Server with Hub Services.

Note: CS3CLNT, RDAAPPC, RDAAPPJ, and JRUNIVP are client applications. Each time you run an EDACFGx procedure, the EDACFGx procedure recreates these members. Therefore, they are valid only for the most recent EDACFGx procedure.

If you do not wish to configure for outbound communications, leave EDAQTCPO and EDAQLU6O commented out.

Syntax

How to Generate Full-Function Server Configuration Files

Important: You must uncomment and supply a value for *all* parameters when you are editing any of the EDAQxxxx procedures. Never leave a parameter blank (=,).

Edit *qualif*.EDALIB.DATA(EDACFGF) as follows and submit the job.

Note: All values to the right of the equal sign (=) should be in uppercase.

```
//*        JOB CARD GOES HERE
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*Purpose: To create the configuration file and test procedures    *
//*         for a Full Function Server                               *
//*                                                                  *
//*Procedure names :                                                 *
//*                                                                  *
//*     FFSDEL   Deletes data sets created by the next step for use *
//*              when re-running the job.  Step is commented out    *
//*              as supplied.                                        *
//*                                                                  *
//*     EDAQCFGF Creates the FFS catalogue used by Server.          *
//*              Creates four data sets and loads them with initial *
//*              data.                                               *
//*                                                                  *
//*     EDAQSERV Creates the following Server files in               *
//*              in qualif.INSTALL.DATA                              *
//*                                                                  *
//*              FFSSERV  - Server configuration file                *
//*              FFSCSG   - Server communications file               *
//*              FFSJCL   - Server JCL                               *
//*              CS3CLNT  - RDAAPP client communications file        *
//*              RDAAPPC  - CLIST to run RDAAPP client test program *
//*              RDAAPPJ  - JCL   to run RDAAPP client test program *
//*              TEMP     - Temporary member used to pass values    *
//*                         between procedures                       *
//*                                                                  *
//*                                                                  *
//** WARNING **  The client members above are re-created each       *
//*              time you run an EDACFGx procedure, so they will    *
//*              only be valid for the last procedure run.           *
//*                                                                  *
//*     EDAQCON  Creates the following Server file  in               *
//*              in qualif.INSTALL.DATA. Also modifies FFSSERV      *
//*              file to include single user (ownerid) of console   *
//*                                                                  *
//*              FFSCON  - Console communications file               *
//*                                                                  *
//*     EDAQIMS  Activates IMS DBCTL access from the server.        *
//*              Modifies FFSSERV and FFSJCL members of              *
//*              qualif.INSTALL.DATA. To complete DBCTL activation  *
//*              the FFSJCL member needs to be edited to complete   *
//*              the FOCPSB DDname statement added by this           *
//*              procedure.                                          *
//*                                                                  *
//*     EDAQTCPI Creates a TCP/IP inbound communications block      *
//*     EDAQLU6I Creates a LU62   inbound communications block      *
```

```
//*              in qualif.INSTALL.DATA(FFSCSG). Uncomment one or   *
//*              more of these procedures depending on the          *
//*              protocol(s) you want the server to listen on       *
//*                                                                 *
//*     The following steps are optional and are only required if   *
//*     you are configuring a Full Function Server with HUB         *
//*     capabilities.                                               *
//*                                                                 *
//*     Each step can be duplicated and different values specified  *
//*     so that multiple outbound node blocks for the same protocol *
//*     can be generated in one run of EDACFGF                      *
//*                                                                 *
//*     Make sure each "NODENAME" specified is UNIQUE               *
//*                                                                 *
//*     EDAQTCPO Creates a TCP/IP outbound node block               *
//*                                                                 *
//*     EDAQLU6O Creates a LU6.2 outbound node block                *
//*                                                                 *
//*                                                                 *
//*Substitutions:                                                   *
//*                                                                 *
//*      qualif  The high level qualifier for the data sets.        *
//*                                                                 *
//*      OWNER   Owner id of the catalogue tables.  May be from     *
//*              one to seven characters long.                      *
//*                                                                 *
//*      DUNIT   Specifies the disk unit to use when allocating the *
//*              data sets.  It is defaulted to SYSDA in this JCL.  *
//*              Should you wish to specify a volume as well as a   *
//*              unit when allocating the data sets, please use the *
//*              following form:                                    *
//*                    DUNIT='myunit,VOL=SER=myvol'                 *
//*              e.g., DUNIT='SYSDA,VOL=SER=USERM1'                 *
//*                                                                 *
//*      SERVTYPE  DO NOT CHANGE THIS VALUE                         *
//*                                                                 *
//*      LICENSEK  The license key that was supplied with the       *
//*                product                                          *
//*                                                                 *
//*      MAXIMUM   The maximum number of users for this Full        *
//*                Function Server                                  *
//*                                                                 *
//*      DEPLOY    The deployment type for this server              *
//*                Values can be                                    *
//*                PRIVATE (default) Each user of the server will   *
//*                                  get a private task which will  *
//*                                  be cleared when they           *
//*                                  disconnect so the task will be *
```

```
//*                                   re-used by the next user.      *
//*                                   NOTE: PDS indexes will not be  *
//*                                   re-read in this mode. This is  *
//*                                   not true for server profiles,  *
//*                                   which are always refreshed for *
//*                                   each user connection.          *
//*                PRIVATE,REREAD     Same as for PRIVATE but all    *
//*                                   PDS files will be re-read for  *
//*                                   each user. Should only be used *
//*                                   for EDADBA service or when in  *
//*                                   development mode.              *
//*                ISOLATED           Each user of the server will   *
//*                                   get a private task which will  *
//*                                   be deleted when they           *
//*                                   disconnect from the server.    *
//*                POOLED             Each user of the server will   *
//*                                   re-use a task and context      *
//*                                   of last user of the task. Task *
//*                                   will NOT be deleted on user    *
//*                                   disconnect.                    *
//*                                                                  *
//*      SYSOUT    SYSOUT class server will use for trace file      *
//*                (When activated)                                  *
//*                                                                  *
//*      APPSNS    If set to YES, a service block called EDAAPPS    *
//*                will be added to FFSSERV member and Application  *
//*                Name spaces will be enabled for that service.    *
//*                The APPROOT setting will have the value of the   *
//*                high level qualifier(s) that were used to unload *
//*                the server tape.                                  *
//*                Default is NO                                     *
//*                                                                  *
//*      CONAPPLD  The VTAM resource name (APPLID) to be used for   *
//*                access to the console                             *
//*                                                                  *
//*      IMSRESLB  The DSN of the IMSRESLB library to be used by    *
//*                the server.                                       *
//*                                                                  *
//*      IMSPZPLB  The DSN of the DFSPZP library in which the PZP   *
//*                load module has been generated.                   *
//*                                                                  *
//*      PZPSUFIX  The 2 character suffix of the PZP module to be   *
//*                loaded from the DFSPZP library.                   *
//*                                                                  *
//*      TCPNAME   The TCP/IP job name. The default is TCPIP for IBM*
//*                For INTERLINK, it the subsystem name.             *
//*                                                                  *
//*      VENDOR    The name of the TCP/IP vendor. Default value is  *
```

```
//*                IBM or it can have a value of INTERLINK or OCS   *
//*                                                                 *
//*      SERVICE   The TCP/IP port number the server will listen on *
//*                                                                 *
//*      HOST      The IP address of the machine that the server    *
//*                is running on.                                   *
//*                                                                 *
//*      CLIENTLU  The VTAM resource name (APPLID) that the test    *
//*                Client will use.                                 *
//*                                                                 *
//*      SERVERLU  The VTAM resource name (APPLID) that the server  *
//*                will listen on.                                  *
//*                                                                 *
//*      MODENAME  The MODENAME that the LU6.2 APPLID was generated *
//*                with.                                            *
//*                                                                 *
//*      LU2APPLD  The LU2 APPLID that the server will listen on    *
//*                                                                 *
//*      NODENAME  This must match the subservers service name if   *
//*                the subserver is running on MVS. For UNIX and NT *
//*                use a 1-8 character descriptive name.            *
//*                                                                 *
//*      SUBHOST   The IP address of the machine that the subserver *
//*                is running on.                                   *
//*                                                                 *
//*      SUBPORT   The port number the subserver is using to        *
//*                listen on.                                       *
//*                                                                 *
//*      LOCALLU   An APPLID used by the server so that it can act  *
//*                as a Client. You can specify '(ddname)' E.G      *
//*                (LU6POOL) which is a ddname pointing to a list   *
//*                of APPLIDs. If you used this option, you will    *
//*                need to ADD a ddname card to the generated server*
//*                JCL that points to the list of APPLIDs           *
//*                                                                 *
//*      PARTLU    The APPLID that the subserver is using to listen *
//*                on.                                              *
//*                                                                 *
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*
//EDAPROCS JCLLIB ORDER=qualif.EDALIB.DATA
//*FFSDEL   EXEC PROC=FFSQDEL,
//*         OWNER=owner_id
//EDAQCFGF EXEC PROC=EDAQCFGF,
//         OWNER=owner_id,
//         DUNIT=SYSDA
//EDAQSERV EXEC PROC=EDAQSERV,
//         SERVTYPE=FFS,
```

```
//          LICENSEK=license key,
//          MAXIMUM=maximum users,
//          DEPLOY=PRIVATE,
//          SYSOUT=sysout class,
//          OWNER=owner_id,
//          APPSNS=NO
//*EDAQCON  EXEC PROC=EDAQCON,
//*          SERVTYPE=FFS,
//*          CONAPPLD=Console applid
//*EDAQIMS  EXEC PROC=EDAQIMS,
//*          SERVTYPE=FFS,
//*          IMSRESLB='IMS RESLIB library name',
//*          IMSPZPLB='IMS PZPLIB library name',
//*          PZPSUFIX=DFSPZP suffix to use
//*EDAQTCPI EXEC PROC=EDAQTCPI,
//*          SERVTYPE=FFS,
//*          TCPNAME=TCPIP,
//*          VENDOR=IBM,
//*          SERVICE=tcp port number,
//*          HOST=host name
//*EDAQLU6I EXEC PROC=EDAQLU6I,
//*          SERVTYPE=FFS,
//*          CLIENTLU=client lu name,
//*          SERVERLU=server lu name,
//*          MODENAME=mode name for APPLID
//*EDAQTCPO EXEC PROC=EDAQTCPO,
//*          SERVTYPE=FFS,
//*          NODENAME=node name of subserver,
//*          TCPNAME=TCPIP,
//*          VENDOR=IBM,
//*          SUBHOST=ip address,
//*          SUBPORT=port number
//*EDAQLU6O EXEC PROC=EDAQLU6O,
//*          SERVTYPE=FFS,
//*          NODENAME=node name of subserver,
//*          LOCALLU=local lu name,
//*          PARTLU=partner lu name,
//*          MODENAME=Vtam mode name
```

where:

qualif

Is the high-level qualifier for the data sets.

owner_id

Is the owner of the catalog. It is usually the TSO Userid of the person configuring the server. For related information, see *Specifying Values for an Ownerid* on page 4-3.

DUNIT

Specifies the disk unit to use when allocating the data sets. The default in this JCL is SYSDA. If you want to specify a volume as well as a unit when allocating the data sets, use the following format:

```
DUNIT='myunit,VOL=SER=myvol'
```

For example,

```
DUNIT='SYSDA,VOL=SER=USERM1'
```

FFS

Is the Full-Function Server setting. Do not change this value.

license key

Is the license key supplied with the product.

maximum users

Is the maximum number of users for this Full-Function Server.

deployment mode

Is the deployment type for this server. The modes of deployment are:

PRIVATE: (default) Each user of the server receives a private task, which will be cleared when they disconnect so the task will be reused by the next user.

Note: PDS indexes will not be reread in this mode. This is not true for server profiles, which are always refreshed for each user connection.

PRIVATE,REREAD: Same as for PRIVATE but all PDS files will be reread for each user. Should be used only for EDADBA service or when in development mode.

ISOLATED: Each user of the server will get a private task, which will be deleted when they disconnect from the server.

POOLED: Each user of the server reuses the task and context of the last user of the task. The server does not delete tasks when the user disconnects.

sysout class

Is the SYSOUT class to which the server sends trace output when tracing is activated. For example, the specification SYSOUT X sends the output to SYSOUT CLASS X.

console applid

Is the VTAM resource name (APPLID) used for Console access.

'IMS RESLB library name'

Is the DSN of the IMSRESLB library to be used by the server.

'IMS PZPLB library name'

Is the DSN of the DFSPZP library in which the PZP load module has been defined.

`DFSPZP suffix`

Is the 2-character suffix of the PZP module that is to be loaded from the DFSPZP library.

`tcp name`

For IBM TCP/IP is the name of the started task or the JOB name for the TCP/IP subsystem address space, if it is different from the default (TCPIP).

For Interlink TCP/IP is the name of the TCP/IP subsystem (no default value).

For OCS use the default value of TCPIP.

Note: The server can support only one TCP/IP vendor at one time for both inbound and outbound blocks. Use multiple servers to receive support from multiple vendors.

`vendor name`

Is the TCP/IP vendor name. Possible values are IBM, INTERLINK, or OCS.

`tcp port number`

Is the TCP/IP port number on which the server listens.

`host name`

Is the IP address of the machine on which the server runs.

`client lu name`

Is the VTAM resource name (APPLID) that the LU6.2 test client uses.

The APPLID for the LU62 test clients must be unique, and cannot be the same as those specified for LU62 inbound communications (`server lu name`). You have the option of using a pooled list of APPLIDs by setting this value to (LU6POOL); for example, CLIENTLU=(LU6POOL). See *DDNAME LUPOOL* on page 4-12 and Chapter 6, *Testing Server Communications*, for more information.

`server lu name`

Is the VTAM resource name (APPLID) on which the server listens for LU6.2 inbound communications. This APPLID must be unique.

`mode name for APPLID`

Is the MODENAME with which the server's LU6.2 APPLID was defined.

The following keywords are for a Full-Function Server with Hub Services. They specify the outbound communications parameters that allow this server to act as a client to the subserver. They configure client node blocks in the EDACSG communications file. If you are not configuring access to a subserver, skip this section and proceed to *Next Steps* on page 4-22.

`node name of subserver`

The name of the service on the subserver to which this server connects. If the subserver is on MVS, this value must match the name of a service specified in the subserver's EDASERVE file. You need to specify a nodename for each outbound communications block. This name must be unique for each block. If the subserver is on UNIX or NT, this value should be a 1-8 character descriptive name. It is used for reference by the CREATE SYNONYM command.

`ip address`

Is the IP address of the machine on which the subserver is running.

`port number`

Is the port name or number on which the subserver is listening.

`local lu name`

Is the APPLID (LU6.2) that the Full-Function Server will use to act as a client to communicate with a subserver.

If you need several client connections to a subserver, you must use 'DDNAME LUPOOL' syntax. See *DDNAME LUPOOL* on page 4-12 and Chapter 6, *Testing Server Communications*, for more information.

`partner lu name`

Is the LU6.2 APPLID on which the subserver listens for inbound communications.

`Vtam mode name`

Is the MODENAME with which the subserver's LU6.2 APPLID was defined.

Next Steps

Once you have completed the configuration of the server, proceed to Chapter 5, *Configuring Server System Features*, to configure the following:

- IBI Subsystem.
- HiperEDA.
- National Language Support.
- Transaction Server for IMS.

If you are *not* configuring any of these features, proceed to Chapter 6, *Testing Server Communications*, to submit the server JCL and test the server communications.

CHAPTER 5

Configuring Server System Features

Topics:

- IBI Subsystem-Based Products
- Installing the IBI Subsystem
- Installing and Configuring HiperEDA
- Configuring the MVS Server for National Language Support
- Configuring a Communications Gateway
- Configuring the Transaction Server for IMS

Several server system features enable you to optimize and customize your environment. These include:

- **The IBI Subsystem**, which provides expanded storage control and error recovery capabilities.
- **HiperEDA**, which uses hiperspaces to store information.
- **National Language Support (NLS)**, which provides a translation component that enables a server to work in all countries and for all languages.
- **Communications Gateway**, which works as an independent server or as a service on a Hub or a Full-Function Server with Hub Services.

Transaction Server for IMS, which provides a means of building on existing applications that perform transaction processing against IMS/TM in order to create new client/server applications that reside on the desktop.

IBI Subsystem-Based Products

The following products use the IBI Subsystem:

- HiperEDA

  The HiperBUDGET feature of HiperEDA uses the IBI Subsystem to regulate use of expanded storage on a system-wide basis.

- FOCUS Database Server

  The FOCUS Database Server uses the IBI Subsystem to communicate between server users and the FOCUS Database Server.

- SmartMode for FOCUS

  SmartMode, which is a service based on a FOCUS database, uses the IBI Subsystem to communicate between server users and sink machines.

- FOCUS IMS BMP

  The IMS BMP extension uses the IBI Subsystem to communicate between server users and the BMP address space.

- Enterprise Control Center Server.

You must install the IBI Subsystem before you install these products.

Benefits of Using the IBI Subsystem

The IBI Subsystem provides the following benefits:

- Expanded storage control.

  The HiperBUDGET facility of HiperEDA allows coordination of MVS hiperspaces across all address spaces running server products, thus allowing comprehensive enforcement of installation policies governing hiperspace usage.

- Enhanced error recovery capability.

  By intercepting task-end and memory termination events, error recovery is enhanced. The subsystem can always perform cleanup when any server product terminates.

- New control facilities.
  - Display status from the MVS system console.
  - Control using the MVS system console or batch jobs.

Note: You must use the subsystem supplied with this release. You cannot use this release of the server with older subsystem code.

How the IBI Subsystem Works

The IBI Subsystem runs in the address space of the server that is using its services. All of the modules used by the subsystem are reentrant and pageable, and reside in the Extended Common Storage Area (XCSA). In addition, all control blocks created by the subsystem reside in the XCSA and are also pageable.

The subsystem uses 20K of XCSA for programs and control blocks, almost all of which is taken by programs.

The subsystem is loaded and initialized by the SUBSYSI utility, which must be run from an APF-authorized library. SUBSYSI can also be used to stop, replace, and/or unload the subsystem modules.

When used for communications between address spaces, the subsystem can use cross-memory services only when the server address space (server or FOCUS database machine) is non-swappable. If cross-memory services cannot be used, then an 8K "mailbox" is allocated in XCSA for each communication path used by the server. The server address spaces use two mailboxes, for a total of 16K; SU address spaces use one mailbox. The subsystem code is run whenever any of the following events occur:

- The server requests subsystem services.
- An operator command is issued with the subsystem name as the first four characters of the command.
- A task-end or memterm event occurs in a server address space.
- The SUBSYSI utility program is run.
- The subsystem supports the CPF macro for prefix reference across a sysplex. For more information on the CPF macro, see your IBI documentation.

Installing the IBI Subsystem

Before starting the installation procedure, be sure that you are familiar with the preliminary information included in the topics *IBI Subsystem-Based Products* on page 5-2, *Benefits of Using the IBI Subsystem* on page 5-2, and *How the IBI Subsystem Works* on page 5-3. For related information about operating and interacting with the IBI Subsystem, see Chapter 10, *IBI Subsystem*.

It is not necessary to update SYS1.PARMLIB(IEFSSN*xx*) to install the IBI Subsystem. The provided SUBSYSI utility dynamically updates the appropriate MVS control blocks to add the IBI Subsystem to the list of installed subsystems.

Even if you update IEFSSN*xx* with the subsystem name, you must still run the SUBSYSI utility with the START option to load and activate the IBI Subsystem.

The SUBSYSI utility requires an APF-authorized library. This library may be an existing library, a member of your system's LNKLST, or a new library created just for SUBSYSI. Whichever option you choose, you may wish to protect SUBSYSI so that it can only be run by authorized personnel.

While installing the IBI Subsystem does not normally require an IPL of your system, an IPL may be needed for changes that were made during installation to take effect (such as adding a new APF-authorized library).

If the IBI Subsystem is installed but not defined in IEFSSN*xx*, the server will abend at startup with an 0C1 in module SSCTL. The second word of the PSW at the point of abend points to the PSA area. For example:

```
PSW at time of abend  '078D0000 80000004'
```

The solution is to define the IBI Subsystem in IEFSSN*xx*. The following is a sample of IEFSSN*xx*:

```
SMS                       /* System Managed Storage        */
JES2,,,Primary            /* JES2                          */
DBDC                      /* IMS DC Component              */
IBIS                      /* IBI subsystem                 */
DSN,DSN3INI,'DSN3EP,-'    /* DB2                           */
```

Note: Appropriate operations and system support staff should be trained in the use of the SUBSYSI utility for starting and controlling the subsystem and the operator commands that the subsystem provides, DISPLAY and SET. In addition, you may need to update your system IPL procedures to ensure the subsystem is properly started after MVS is initialized.

Procedure How to Install the IBI Subsystem

Follow these steps to install the IBI Subsystem. All of the samples discussed are located in *qualif*.EDALIB.DATA.

1. Choose a valid subsystem name. It must be four characters long. The default name is IBIS.

2. Choose an APF-authorized library. SUBSYSI must be run from an APF-authorized library. You may copy *qualif*.EDALIB.LOAD(SUBSYSI) to a library in the LNKLST or to an existing authorized library, or you may create a new library for this purpose.

   As only authorized personnel should run SUBSYSI, it is recommended that the library in which it resides be protected from unauthorized access.

   While *qualif*.EDALIB.LOAD is APF-authorized when your server is installed, you will probably want to use a separate library in order to help protect SUBSYSI from unauthorized use.

3. Update SYS1.PARMLIB (optional). If you need to create a new APF-authorized library, or are adding non-swappable programs (TSCOM3 for EDA), update SYS1.PARMLIB as needed to specify these changes.
4. Zap to change the subsystem name (optional). If you have chosen a name other than IBIS for the IBI Subsystem, edit the JCL in *qualif*.EDALIB.DATA(SUBSYSNM) with the appropriate name and run it to apply the zap.
5. Copy modules to an APF-authorized library and protect the library. The JCL in *qualif*.EDALIB.DATA(SUBSYSCP) will copy SUBSYSI and related modules from *qualif*.EDALIB.LOAD into an APF-authorized library. This library may be a library dedicated to SUBSYSI, a member of the LNKLST, or any other library you choose. As authorized personnel should only run SUBSYSI, ensure that the library is protected from unauthorized access.
6. Protect modules or delete them from *qualif*.EDALIB.LOAD. SUBSYSI and the other modules that were copied should be deleted from *qualif*.EDALIB.LOAD or protected so that they may only be executed by authorized personnel.
7. Set up automatic start of the subsystem (optional). If desired, set up JCL to run SUBSYSI and initialize the IBI Subsystem at system startup. This is especially useful if HiperEDA is installed, to ensure that all users are subject to the HiperBUDGET controls established by the subsystem. Sample JCL to run SUBSYSI is located in members SUBSYSIJ and SUBSYSIH of *qualif*.EDALIB.DATA.
8. Set up a SUBSYSI cataloged procedure (optional). If desired, set up a cataloged procedure so SUBSYSI can be run as a started task. A sample procedure is located in SUBSYSP.
9. IPL the system (optional). The system must be IPLed to activate changes to SYS1.PARMLIB. If an authorized library was already available, the IPL is not necessary in order to use the subsystem, although some facilities, such as non-swappable servers, will not be available until after the IPL.
10. Start the IBI Subsystem. Bring up the subsystem by running SUBSYSI with the START parameter. It is now ready for use.

Procedure How to Maintain the IBI System

After maintenance has been applied to your server:

1. Re-run all of the appropriate zap and copy jobs.
2. Run SUBSYSI with the REPLACE parameter to load the new subsystem modules.

   Ensure that no servers are using the subsystem when the REPLACE operation is performed.

Procedure How to Uninstall the IBI Subsystem

You may uninstall the IBI Subsystem by running SUBSYSI with the REMOVE parameter. When doing this, be sure that no server address spaces that use the IBI Subsystem are active.

Installing and Configuring HiperEDA

To take advantage of the HiperEDA and HiperBUDGET facilities, you must install and configure them. For details, see *HiperEDA Configuration* on page 5-6 and *HiperBUDGET and the IBI Subsystem* on page 5-8.

Hardware and Software Requirements for HiperEDA

The HiperEDA installation can be performed by anyone with a working knowledge of MVS. Some server knowledge is required to restrict options or resource usage. The server must be installed prior to installing HiperEDA.

The hardware and software requirements for the HiperEDA option are:

- MVS/ESA V3 or higher.
- MVS/DFP V3.1 or higher.
- IBM ES/3090S, ES/3090J, ES/3090JH, ES/3090T, any ES/9000 model 9021, or an Amdahl or Hitachi compatible.

If you are installing HiperEDA with MVS/ESA V4.2.0, 4.2.2, or V4.3.0, contact IBM for APAR OY61461.

HiperEDA Configuration

You can activate and customize HiperEDA using SET commands in EDASPROF. You must inform the server that HiperEDA has been installed by adding SET HIPERINSTALL=ON to the EDASPROF entries and accept the defaults. However, because there would be no limit to the number of hiperspaces that the server can create, we strongly recommend that you set limits using the SET parameters listed below, as well as those described in *HiperBUDGET and the IBI Subsystem* on page 5-8. Together, they comprise the HiperBUDGET facility of HiperEDA.

The following is a list of the SET parameters available for HiperEDA.

```
SET HIPERINSTALL={ON|OFF}
```

where:

`ON`

Installs HiperEDA.

`OFF`

Disables HiperEDA. This is the default.

`SET HIPERSPACE=`*`nnn`*

where:

nnn

Is the number of (4K) pages for aggregate hiperspace size. By default there is no limit. This parameter is equivalent to SET TCBLIM=*nnn* under the IBI Subsystem (see *HiperBUDGET and the IBI Subsystem* on page 5-8). If both parameters are used and different limits are set, then the lower limit is enforced.

`SET HIPERFILE=`*`nnn`*

where:

nnn

Is the maximum number of (4K) pages for an individual hiperspace size. The default is 524,288 (4K) pages or 2 GB in size. This parameter is equivalent to SET FILELIM=*nnn* under the IBI Subsystem (see *HiperBUDGET and the IBI Subsystem* on page 5-8). If both parameters are used and different limits are set, then the lower limit is enforced.

`SET HIPERCACHE=`*`nnn`*

where:

nnn

Is the number of hipercache (4K) pages. The default is 256 (4K) pages or 1M in size.

`SET HIPEREXTENTS=`*`nnn`*

where:

nnn

Is the number of extend operations to be performed. The default number is 127.

`SET HIPERLOCKED={ON|OFF}`

where:

`ON`

Disallows processing of user interface commands.

`OFF`

Allows processing of user interface commands (such as SET HIPEREDA=ON or SET HIPEREDA=OFF). This value is the default.

```
SET HIPEREDA={ON|OFF}
```

where:

ON

Activates a HiperEDA session after HiperEDA has been installed. This value is the default.

OFF

Deactivates a HiperEDA session.

Note: All of the SET options described above (except for SET HIPEREDA) can only be executed from EDASPROF.

HiperBUDGET and the IBI Subsystem

The IBI Subsystem is a replacement for the IBI SVC. It provides communication among address spaces running server products on the same MVS system. HiperBUDGET uses this subsystem to regulate and report the overall use of hiperspace on that MVS system. It accomplishes this through the enforcement of predefined limits on hiperspace consumption at the system, server, user, and file levels; limits set at the lower levels cannot exceed those set at a higher level. You can set these limits for the IBI Subsystem. For an illustration, see *Setting Limits for the IBI Subsystem* on page 5-8.

For more information, see Chapter 10, *IBI Subsystem*.

Example Setting Limits for the IBI Subsystem

The following SET commands are issued from the TSO command line. The examples use IBIS as the subsystem name.

```
[/IBIS] SET MVSLIM={nnn|-1}
```

where:

nnn

Is the maximum number of (4K) pages of hiperspace for all server products on that MVS system.

-1

Specifies that there is no limit checking.

```
[/IBIS] SET SERVLIM={nnn|-1}
```

where:

nnn

Is the maximum number of (4K) pages of hiperspaces allowed for multiple users on a per server basis.

`-1`

Specifies that there is no limit checking.

`[/IBIS] SET TCBLIM={nnn|-1}`

where:

nnn

Is the maximum number of (4K) pages available for each individual user.

`-1`

Specifies that there is no limit checking.

This parameter is equivalent to SET HIPERSPACE=*nnn*, described in *Installing and Configuring HiperEDA* on page 5-6. If both parameters are specified, then the smaller value takes precedence.

`[/IBIS] SET FILELIM={nnn|-1}`

where:

nnn

Is the maximum number of (4K) pages of hiperspace on an individual file basis.

`-1`

Specifies that there is no limit checking.

This parameter is equivalent to SET HIPERFILE=*nnn* under the IBI Subsystem that is discussed in *Installing and Configuring HiperEDA* on page 5-6. If both parameters are specified, then the smaller value takes precedence.

Configuring the MVS Server for National Language Support

You can customize your NLS server configuration to:

- **Change your code page settings.** You can change the code page settings on the server by editing the code page generation list file (CPCODEPG) and then running the Transcoding Services Generation Utility (TSGU) to generate the new transcoding table. The new transcoding table is based on the list of code pages defined in the updated code page generation list file (CPCODEPG). For more information, see *Changing the MVS Server Code Page Settings* on page 5-11.
- **Customize NLS default characteristics.** You can change the server code page setting and the language the server uses for error messages by editing the NLS configuration file (NLSCFG). For more information, see *Configuring MVS Server NLS Default Characteristics* on page 5-16.

- **Customize monocasing.** You can customize the monocasing table by editing the code page definition file (CP*NNNNN*) and then generating the new monocasing table file (CASETBL) with the TSGU. For more information, see *Configuring Customized MVS Server NLS Monocasing* on page 5-19.
- **Customize sorting.** You can customize the sorting table by editing the code page definition file (CP*NNNNN*) and then generating the new sorting table file (SORTTBL) with the TSGU. For more information, see *Configuring for Customized MVS Server NLS Sort Sequences* on page 5-21.

Note: Customization of your NLS configuration is only necessary if your enterprise requires support for alternate or additional code pages, or custom monocasing and sorting.

Reference

NLS Configuration Files for MVS

The following table describes the NLS configuration files:

Member	Data Set	Description
casetbl	*qualif*.EDANLS.DATA	Monocasing table. This file contains the monocasing tables that are generated using the TSGU procedure and are based on the code page generation list (CPCODEPG).
cpcodepg	*qualif*.EDACTL.DATA	Code page generation list. This file is a list of currently active code pages. It contains one code page for the server and multiple code pages for the client.
CP*NNNNN*	*qualif*.EDACTL.DATA	Code page definition file. This file contains information on each code point value in the code page. There is a code page definition file for every code page listed in the known code page file (CPXCPTBL).
cpxcptbl	*qualif*.EDACTL.DATA	Known code page list. This file contains a list of enabled code pages for iWay Software's clients and servers.
langtbl	*qualif*.EDAMSG.DATA	Known language list file. This file contains a list of enabled language names for iWay Software's clients and servers.

Member	Data Set	Description
nlscfg	*qualif*.EDAMSG.DATA	NLS configuration file. This file controls the server code page setting and the language the server uses for error messages. There is an NLSCFG file for both the client and server.
CS3CLNT	*qualif*.INSTALL.DATA	Communications configuration file. This file contains the code page setting and DBCS setting for the client.
sortTbl	*qualif*.EDANLS.DATA	Sorting table. This file contains the sorting tables which are generated using the TSGU and are based on the contents of the code page generation list (cpcodepg).
trantbl	*qualif*.EDANLS.DATA	Transcoding table file. This file contains the transcoding tables that are generated using the TSGU and are based on the contents of the code page generation list (CPCODEPG).

Changing the MVS Server Code Page Settings

You will need to perform the following steps to change your code page configuration:

1. Identify the current code page settings.
2. Identify alternate or additional code pages by referencing the known code page file (CPXCPTBL).
3. Edit the code page generation list file (CPCODEPG) with the code page information collected from the known code page file (CPXCPTBL).
4. Generate the new transcoding tables based on the updated code page generation list file (CPCODEPG) using the TSGU.
5. If you generated a new server code page in step4, you will need to modify the NLS configuration file (NLSCFG) to identify the new server code page and language used by the server for error messages.

   Note: This step is only necessary if you generated a new server code page in step 4. If you generated new client code pages, you do not need to modify the NLS configuration file (NLSCFG).

Procedure Identifying Current MVS Code Page Settings (Step 1 of 5)

Code page settings are reflected in the code page generation list file *qualif*.EDACTL.DATA(CPCODEPG). This file contains code page settings for both client and server and is used to generate the transcoding tables with the TSGU. To identify your current code page settings, view the code page generation list file (cpcodepg).

During server installation, your server is set up by default with the US EBCDIC code page 37. The client code page setting on the server is set up to support the following default code pages:

- US Mainframe EBCDIC CP 37
- US PC ASCII CP 437
- PC Windows ANSI CP 137
- Mainframe Latin-1 1047
- Unicode (UTF-8) 65001

Example Using the Code Page Generation List File (CPCODEPG) for MVS

The following is an example of a code page generation list file (CPCODEPG).

```
CP00037   E SBCS  US IBM MF EBCDIC code
CP00437   A SBCS  US PC ASCII code
CP00137   A SBCS  ANSI Character Set for MS-Windows
CP01047   E SBCS  IBM MF Open Systems (Latin 1)
CP65001   A UTF8  Unicode (UTF-8)
```

Procedure Identifying Alternate/Additional MVS Code Pages (Step 2 of 5)

If your server requires support for alternate or additional code pages, you can identify the code pages by referencing the known code page file *qualif*.EDACTL.DATA (CPXCPTBL).

Note: If the desired code page is not listed in the known code page file (CPXCPTBL), refer to the appropriate IBM Character Data Representation Architecture (CDRA) document and create your own, or contact your local iWay Software representative for information about additional code pages.

Syntax How to Use the Known Code Page File (CPXCPTBL) for MVS

`CPnnnnn b dbcs-id description`

where:

`CP`

Is the code page prefix (always CP).

`nnnnn`

Is the code page number.

`b`

Is the character type. Possible values are:

`A` for ASCII.

`E` for EBCDIC.

`dbcs-id`

Is the DBCS identifier (DBCSID).

`description`

Is a description of the code page.

Example Using the Known Code Page File (CPXCPTBL) for MVS

The following is a listing of the contents of the known code page file (CPXCPTBL):

```
CP00037   E SBCS  US IBM MF EBCDIC code
CP00437   A SBCS  US PC ASCII code
CP00137   A SBCS  ANSI character set for MS-Windows
CP00500   E SBCS  IBM MF International European
CP00273   E SBCS  IBM MF Germany F.R./Austria
CP00277   E SBCS  IBM MF Denmark, Norway
CP00278   E SBCS  IBM MF Finland, Sweden
CP00280   E SBCS  IBM MF Italy
CP00284   E SBCS  IBM MF Spain/Latin America
CP00285   E SBCS  IBM MF United Kingdom
CP00297   E SBCS  IBM MF France
CP00424   E SBCS  IBM MF Israel(Hebrew)
CP00850   A SBCS  IBM PC Multinational
CP00856   A SBCS  IBM PC Hebrew
```

```
CP00860   A SBCS  IBM PC Portugal
CP00862   A SBCS  IBM PC Israel
CP00863   A SBCS  IBM PC Canadian French
CP00865   A SBCS  IBM PC Nordic
CP00637   A SBCS  DEC Multinational Character Set
CP00600   A SBCS  DEC German NRC Set
CP00604   A SBCS  DEC British ( United Kingdom ) NRC Set
CP00608   A SBCS  DEC Dutch ( Netherland ) NRC Set
CP00612   A SBCS  DEC Finnish ( Finland ) NRC Set
CP00616   A SBCS  DEC French ( Flemish and French/Belgian ) NRC Set
CP00620   A SBCS  DEC French Canadian NRC Set
CP00624   A SBCS  DEC Italian ( Italy ) NRC Set
CP00628   A SBCS  DEC Norwegian/Danish ( Norway,Denmark ) NRC Set
CP00632   A SBCS  DEC Portugal NRC Set
CP00636   A SBCS  DEC Spanish ( Spain ) NRC Set
CP00640   A SBCS  DEC Swedish ( Sweden ) NRC Set
CP00644   A SBCS  DEC Swiss ( Swiss/French and Swiss/German ) NRC Set
CP00930   E SOSI  Japanese IBM MF Katakana Code Page (cp290+cp300)
CP00939   E SOSI  Japanese IBM MF Latin Extended (cp1027+cp300)
CP00932   A SJIS  Japanese PC Shift-JIS (cp897+cp301)
CP00942   A SJIS  Japanese PC Shift-JIS Extended (cp1041+cp301)
CP10942   A EUC   Japanese PC EUC
CP10930   E SOSIF Japanese Mainframe ( Fujitsu )
CP20930   E SOSIH Japanese Mainframe ( Hitachi )
CP00933   E SOSI  Korean IBM MF Extended (cp833+cp834)
CP00934   A KPC   Korean IBM PC (cp891+cp926)
CP00944   A KPC   Korean IBM PC Extended (cp1040+cp926)
CP00949   A KKS   Korean KS5601 code (cp1088+cp951)
CP00935   E SOSI  PRC IBM MF Extended (cp836+cp837)
CP00937   E SOSI  Taiwanese IBM MF (cp37+cp835)
CP00948   A TPC   Taiwanese IBM PC Extended (cp1043+cp927)
```

```
CP10948   A TBIG5 Taiwanese PC BIG-5 code
CP20948   A TNS   Taiwanese PC National Standard code
CP30948   A TTEL  Taiwanese PC Telephone code
CP00857   A SBCS  IBM PC Latin 5 (Turkish)
CP00920   A SBCS  ISO-8859-9 (Latin 5, Turkish)
CP01026   E SBCS  IBM MF Turkish
CP01252   A SBCS  ISO-8859-1 (Latin 1)
CP00819   A SBCS  ISO-8859-1 (Latin 1, IBM Version 8X/9X undef'ed)
CP00912   A SBCS  ISO-8859-2 (Latin 2)
CP01047   E SBCS  IBM MF Open Systems (Latin 1)
CP65001   A UTF8  Unicode (UTF-8)
CP00916   A SBCS  ISO-8859-8 (Hebrew)
```

Procedure **Adding Alternate or Additional MVS Code Pages (Step 3 of 5)**

The code page generation list file *qualif*.EDACTL.DATA(CPCODEPG) is a list of code pages to be generated with the TSGU. It must be changed manually, using an editor, to contain the additional or alternate set of code page numbers identified in the known code page file (CPXCPTBL) (step 2 of 5).

Add the information for each additional or alternate code page on a separate line, one for the server and one or more for the clients accessing the server. You can copy the code page information from the known code page file (CPXCPTBL) into the code page generation list file (CPCODEPG). Note that only CP and the code page number (for example, CP00037) are required to generate the new transcoding tables. The maximum number of code page entries in the file is sixteen.

Procedure Generating Transcoding Tables Using the TSGU (Step 4 of 5)

The Transcoding Services Generation Utility (TSGU) program is used to generate the NLS transcoding table file *qualif*.EDANLS.DATA(TRANTBL), based on the list of code pages defined in the code page generation list file (CPCODEPG). The transcoding table file (TRANTBL) contains the transcoding tables for all the possible code page combinations. You can generate a new transcoding table with the TSGU by completing the following steps:

1. The sample JCL to execute TSGU is located in *qualif*.EDACTL.DATA(TSGUJCL). Edit this JCL stream to conform to your site requirements by replacing *qualif* with the appropriate value and supplying a valid JOB card.

   The output of TSGU is the transcoding table file (TRANTBL), which is based on the contents of the code page generation list file (CPCODEPG). The TRANTBL is generated in *qualif*.EDANLS.DATA. This data set must be allocated in the server's JCL to the DDname ERRORS.

2. If you get a duplicate GCGID, choose which one to discard and replace it with eight periods as follows:

   .....

3. The following confirms the generated information.
   - INFO shows basic NLS information.
   - INFO CP shows a list of translation tables.

Procedure Changing the MVS Server Code Page Setting (Step 5 of 5)

If you generate a new server code page in step 4 and want the server to recognize this new code page, you will need to modify the NLS configuration file *qualif*.EDAMSG.DATA(NLSCFG) to identify the new server code page and language used by the server for error messages. This step is only necessary if you generated a new server code page in step 4. If you only generated new client code pages, you do not need to modify the NLS configuration file (NLSCFG). For details on changing the NLS configuration file, see *Configuring MVS Server NLS Default Characteristics* on page 5-16.

Configuring MVS Server NLS Default Characteristics

You can change the server code page and the language the server uses for error messages by editing the NLS configuration file *qualif*.EDAMSG.DATA(NLSCFG). You can reference the known language list file *qualif*.EDAMSG.DATA(LANGTBL) for a list of enabled language names. For more information on the NLS configuration file (NLSCFG), see Appendix A, *TSGU and NLS Configuration File*.

Error messages are only translated for German, Spanish, Swedish, French, Dutch, and Japanese. Error messages for all other languages are translated to English.

Note: Swedish and some other languages may only be partially translated.

Example Changing the MVS Server Code Page and Language

You can change the server code page and language of the server by editing the LANG setting in the NLS configuration file *qualif*.EDAMSG.DATA(NLSCFG). Changing the LANG setting sets all other parameters to valid values.

Note: The LANG setting should match the three-character language abbreviation (language ID) or the language name from the known language list file qualif.EDAMSG.DATA(LANGTBL).

The following example illustrates how you can change English to Dutch.

```
LANG = AMENGLISH
```

to

```
LANG = DUT
```

Syntax How to Use the Known Language List File (LANGTBL) to Identify Enabled Language Names for MVS

Enabled language names are contained in the known language list file *qualif*.EDAMSG.DATA(LANGTBL).

```
mmm aaaaaaaaaaaa bbb nnnnn cc
```

where:

mmm

Is a language code (the international dial code).

aaaaaaaaaaaa

Is the language name. Twelve characters is the maximum.

bbb

Is a three-character language abbreviation (language ID) that is used in the LANG setting.

nnnnn

Is the code page number.

cc

Is the hexadecimal value of the dollar symbol.

Example Using the Known Language List File (LANGTBL) for MVS

The following is an example of the contents of the known language list file *qualif*.EDAMSG.DATA(LANGTBL):

```
001AMENGLISH   00037
001ENGLISH     00037
045DANISH      DAN0027767
031DUTCH       DUT00037
358FINNISH     FIN0027867
033FRENCH      FRE00297
049GERMAN      GER00273
049GERMAN      GE500500
972HEBREW      HEB00424
039ITALIAN     ITA00280
081JAPANESE    JPN00930E0
081JAPANESE    JPE00939
082KOREAN      KOR00933
047NORWEGIAN   NOR0027767
351PORTUGUESE  POR00037
034SPANISH     SPA00284
046SWEDISH     SWE0027867
086T-CHINESE   ROC00937
090TURKISH     TUR01026AD
044UKENGLISH   UKE002854A
```

Configuring Customized MVS Server NLS Monocasing

Monocasing (also called Case Conversion) is the conversion of a letter from its lowercase to uppercase form (or vice versa). As part of the basic server initialization, the server is set up with standard monocasing where all requests, except for data between single quotes, are converted to uppercase according to the monocasing table (CASETBL). The monocasing table file *qualif*.EDANLS.DATA(CASETBL) converts a-z to A-Z only. If you require customized monocasing, such as special upper/lowercase accented characters, then you must modify the code page definition file (CP*NNNNN*) and then generate a new NLS monocasing table file (CASETBL) using the TSGU. The new monocasing table is based on the changes made to the code page definition file (CP*NNNNN*).

Note: The server's user functions, LOCASE and UPCASE, respect the NLS monocasing table file (CASETBL).

Procedure **How To Customize Your NLS Monocasing Table for MVS**

Note: As part of the basic initialization, monocasing tables are provided for most of the common European languages. You will only need to customize the monocasing tables if you require a special monocasing configuration.

NLS monocasing involves language-sensitive (code page sensitive) uppercase and lowercase conversion. You can customize each character's attributes by completing the following steps:

1. Edit the code page definition file (CP*NNNNN*), which is named by the code page number. For example, CP00037 contains the monocasing information for US English code page 37. For more information on the code page definition file (CP*NNNNN*), see *How to Use the Code Page Definition File (CPNNNNN) for MVS* on page 5-23.

   Note: You can reference the known code page file (CPXCPTBL) to find the name of the code page definition file.

   The code page definition file (CP*NNNNN*) contains the Code Point and Graphic Character Global Identifier (GCGID). Make the appropriate changes to GCGID uppercase in the third column and Character type in the fifth column.

The following is a chart of a sample code page definition file layout:

Code Point	GCGID	GCGID uppercase	Sort weight	Character type
00	..NUL...	--------	00	-
01	SS000000	--------	01	-
02	SS010000	--------	02	-
÷				
40	SM050000	--------	40	S
41	SS000000	--------	41	U
42	SS000000	--------	42	U
÷				
61	LA020000	LA010000	61	L
62	LB020000	LB010000	62	L
÷				
F1	ND010000	--------	61	D
F2	ND020000	--------	62	D
÷				
FF	SP300000	--------	FF	-

2. Edit the code page generation list file (CPCODEPG) and add the code page definition information as described in *Changing the MVS Server Code Page Settings* on page 5-11. The updated code page generation list (CPCODEPG) is used to regenerate the custom monocasing table file (CASETBL).

3. Execute the TSGU with the parameter CASE. The sample JCL is located in *qualif*.EDACTL.DATA(TSGUJCL).

   The TSGU generates the updated NLS monocasing table file *qualif*.EDANLS.DATA (CASETBL).

Note: For additional information on modifying monocasing values in the code page definition file, refer to the IBM CDRA Library SC09-9391-00 Level 1 Registry or contact your local iWay Software representative.

Configuring for Customized MVS Server NLS Sort Sequences

As part of the basic server initialization, the server is set up with standard sorting. Standard sorting causes the server to use sort sequences of the binary representation of a character string according to the sorting table *qualif*.EDANLS.DATA(SORTTBL). If you require customized sorting, such as changing your sort order to account for Swedish umlauts (Ü), then you must modify the NLS sorting table file (SORTTBL).

Procedure How to Customize Your NLS Sort Tables for MVS

Note: As part of the basic initialization, sorting tables are provided for most of the common European languages. You will only need to customize the sorting tables if you require a special sorting sequence.

If you want to use a weighted sort that accounts for characters that are out of binary sequence, you can customize the sort tables by completing the following steps.

1. Edit the code page definition file qualif.EDACTL.DATA(CPNNNNN), which is named by the code page number. For example, CP00037 contains the sorting information for US English code page 37. For more information on the code page definition file (CPNNNNN), see *How to Use the Code Page Definition File (CPNNNNN) for MVS* on page 5-23.

   Note: You can reference the known code page file (CPXCPTBL) to find the name of the code page definition file.

   The code page definition file (CPNNNNN) contains the Code Point and Graphic Character Global Identifier (GCGID). Make the appropriate changes to the Sort weight listed in the fourth column.

The following is a chart of a sample code page definition file layout:

Code Point	GCGID	GCGID uppercase	Sort weight	Character type
00	..NUL...	--------	00	-
01	SS000000	--------	01	-
02	SS010000	--------	02	-
÷				
40	SM050000	--------	40	S
41	SS000000	--------	41	U
42	SS000000	--------	42	U
÷				
61	LA020000	LA010000	61	L
62	LB020000	LB010000	62	L
÷				
F1	ND010000	--------	61	D
F2	ND020000	--------	62	D
÷				
FF	SP300000	--------	FF	-

2. Edit the code page generation list file (CPCODEPG) and add the code page definition information as described in *Changing the MVS Server Code Page Settings* on page 5-11. The updated code page generation list (CPCODEPG) is used to regenerate the custom sorting table file (SORTTBL).

3. Execute the TSGU with the parameter SORT. The sample JCL is located in qualif.EDACTL.DATA(TSGUJCL).

   The TSGU generates the updated NLS sorting table file qualif.EDANLS.DATA (SORTTBL).

Note: For additional information on modifying monocasing values in the code page definition file, refer to the IBM CDRA Library SC09-9391-00 Level 1 Registry or contact your local iWay Software representative.

Syntax

How to Use the Code Page Definition File (CPNNNNN) for MVS

The code page definition file *qualif*.EDACTL.DATA(CP*NNNNN*) contains information on the characters for each code point value in the code page.

dd aaaaaaaa aaaaaaaa xx h

where:

dd

Is the hexadecimal code point value (00 through FF).

aaaaaaaa

Is the Graphic Character Global Identifier (GCGID).

xx

Is the Sort weight. Consists of a hexadecimal code point value (00 through FF).

h

Is the character type. Possible values are:

- L for Lower-case alphabet.
- U for Upper-case alphabet.
- A for Asian (non-alphabet) character.
- D for Digit.
- S for Special character.
- C for Control (non-printable) character.

Modifying the Server JCL

Concatenate *qualif*.EDANLS.DATA as the first data set of ddname ERRORS in the server JCL. This causes the server to use the NLS files in *qualif*.EDANLS.DATA.

Customizing MVS Client NLS Services

During the client installation, your client is set up by default with the US EBCDIC code page 37. Customization of your NLS configuration is only necessary if your enterprise requires support for an alternate client code page. Client code page customization involves editing the NLS configuration file *qualif*.EDAMSG. DATA(NLSCFG).

Reference

Changing Your MVS Client Code Page Setting

You can change your client code page settings by manually editing the NLS configuration file *qualif*.EDAMSG.DATA(NLSCFG). The client code page settings in the NLS configuration file are sufficient for all client connections. However, if you wish, you can add code page and DBCS information in the communications configuration file *qualif*.INSTALL.DATA(CS3CLNT) for documentation purposes.

Example Using the NLS Configuration File (NLSCFG) for MVS

You can use an editor to change the client code page and language the client uses for error messages by editing the NLS configuration file. Changing the LANG setting sets all other parameters to valid values. The following is an example of a typical NLS configuration file:

```
LANG = AMENGLISH
```

Syntax How to Use the Communications Configuration File (CS3CLNT) for MVS

The NLS settings for CODE_PAGE and DBCS are located in the communications block of the communications configuration file *qualif*.INSTALL.DATA(CS3CLNT).

`CODE_PAGE`

Is the client code page. The hard-coded default value is 37.

`DBCS`

DBCS ID.

Example Changing the Communications Configuration File (CS3CLNT) for MVS

The following is an example of the contents of the code page settings in a typical communications configuration file *qualif*.INSTALL.DATA(CS3CLNT).

```
BEGIN
  PROTOCOL =  TCP
  HOST =      IBIMVS
  SERVICE =   1234
  CLASS =     CLIENT
  CODE_PAGE = 500
END
```

Configuring a Communications Gateway

You can configure a communications gateway as an independent server or as a service on a Hub or a Full-Function Server with Hub Services. On a server for MVS, the communications gateway is called EDAGATE.

EDAGATE is a protocol router. It receives (inbound) client requests using one or more protocols and routes the requests (outbound) using a different protocol. To configure the server for routing, both the client and the server must be using the same protocol. See Chapter 8, *Modifying the Server Configuration*, for more information about matching communications values.

Note: If you did not configure outbound communications when you generated your server configuration files, see Chapter 8, *Modifying the Server Configuration*, for information on how to add an outbound communications block to your communications configuration file.

Procedure How to Configure for EDAGATE

To configure for EDAGATE, add a service block to the EDASERVE configuration file, modify it for the server, and copy the EDAGATE sample file to the EDACSG communications configuration file.

1. If you are using LU6.2 or TCP/IP from the client to EDAGATE, copy the file *qualif*.EDACTL.DATA(CS3GWY), as shown below, into EDASERVE.

```
*****************************************************************
**                SERVICE for EDAGATE                          **
*****************************************************************
SERVICE                 = GATE1
PROGRAM                 = EDAGATE
MAXIMUM                 = nnn
SERVINIT                = *,++
  DYNAM ALLOC FILE STDOUT   SYSOUT ? LRECL 132 BLKSIZE 132 RECFM F
```

   where:

   GATE1

   Is the name of the EDAGATE service. If you are using LU6.2, or TCP/IP, the service name is GATE1.

   nnn

   The maximum number of users allowed for the service.

   ?

   The SYSOUT class to which output will be directed. For example, the specification SYSOUT X sends the output to SYSOUT CLASS X.

2. Copy the file *qualif*.EDACTL.DATA(EDAGATE), as shown below, into the EDACSG communications configuration file. Copy this code after the NAME global block and before the first NODE block. See the example, *MVS Client Connecting to an EDAGATE Server* on page 5-27.

```
GATEWAY = GATE1
BEGIN
  ROUTE = node1
  ROUTE = node2
END
```

   where:

   node1, node2

   Are the values for the NODE keyword for the client node blocks that connect to the subservers. If you did not configure an outbound block in the EDACFGx procedure, you can add them later. See Chapter 8, *Modifying the Server Configuration*, for information on manually adding communication nodes.

If you are configuring only one router, delete ROUTE=*node2*. If you are configuring more than two routers, add additional ROUTE keywords as necessary.

Note:

- You can have more than 1 route value for a given gateway block:

```
GATEWAY = GATE1
 BEGIN
   ROUTE = node1
   ROUTE = node2
   ROUTE = node3
 END
```

- You can have more than 1 gateway block in a single EDACSG file:

```
GATEWAY = GATE1
 BEGIN
   ROUTE = node1
 END
GATEWAY = GATE2
 BEGIN
   ROUTE = node2
 END
```

Example MVS Client Connecting to an EDAGATE Server

Client	Server

EDACSG
`NAME = MVS Client`
`QUEUE DEPTH = 8`
`;TRACE = 3`
`NODE = MYSERVE`
`BEGIN`
`  PROTOCOL = LU62`
`  CLASS = CLIENT`
`  LOCAL LU NAME = ED38APP2`
`  PARTNER LU NAME = ED38APP1`
`  TP NAME = MVSSRVR`
`  MODE NAME = PARALLEL`
`  ;TRACE = 3`
`END`
	EDASERVE
	`SERVICE = GATE1`
	`PROGRAM = EDAGATE`
	`.`
	`.`
	`.`
	EDACSG
	`NAME = MVS Server`
	`QUEUE DEPTH = 8`
	`;TRACE = 3`
	`GATEWAY = GATE1`
	`BEGIN`
	`  ROUTE = MYSERVE`
	`END`
	`NODE = LU6IN`
	`BEGIN`
	`  PROTOCOL = LU62`
	`  CLASS = AGENT`
	`  LOCAL LU NAME = ED38APP1`
	`  TP NAME = MVSSRVR`
	`  ;TRACE = 3`
	`END`
	`NODE = MYSERVE`
	`BEGIN`
	`  PROTOCOL = TCP`
	`  CLASS = CLIENT(USERVE)`
	`  HOST = 123.45.12.34`
	`  SERVICE = 3333`
	`  ;TRACE = 3`
	`END`

The following client and server values must match:

- The client node value must match the value for ROUTE in the server's EDACSG.
- The client node value must match the value for NODE in the outbound node block in the server's EDACSG. The client node value must not match a service name in EDASERVE. If the client node does match a service name or an outbound block node name and EDAGATE is active, then the client receives a -12 error message.

Note: You cannot use CLASS = CLIENT(*outbound block*) from the client. You can use CLASS = CLIENT(*subserve*) on the router.

Reference Security Issues for EDAGATE

The following security issues apply when using EDAGATE:

- Already Verified (Trusted Mode) works.
- Pooled deployment is not valid.
- There are three components involved: client, router, and subserver. When the client attempts to connect to the router, the router will verify whether external security is on. If external security is off, the router will not check the client's security.
- The subserver will check security if security is on. The subserver will know the node that originated the request and allow Already Verified Security to be used if appropriate.
- If you are configuring the server as either a Full-Function Server with EDAGATE or a Hub Server with EDAGATE, and do not want the router to verify security, then add the line

  ```
  EXTSEC = OFF
  ```

  to the Service block for the router directly after SERVICE=GATE1. For example:

  Server (EDASERVE)

  ```
  SERVICE    = GATE1
  EXTSEC     = OFF
  PROGRAM    = EDAGATE
                 .
                 .
  ```

Configuring the Transaction Server for IMS

The Transaction Server for IMS allows any client on any platform to invoke a transaction running under the control of an IMS/TM transaction-processing monitor. It provides a means of building on existing applications that perform transaction processing against IMS/TM, to create new client/server applications that reside on the desktop.

To call an IMS transaction, execute the CALLIMS function from a Dialogue Manager procedure. The function CALLIMS invokes an IMS/TM transaction through the use of APPC/IMS. CALLIMS requires IMS Release 4.1 (IMS/TM) and APPC/MVS.

CALLIMS Processing

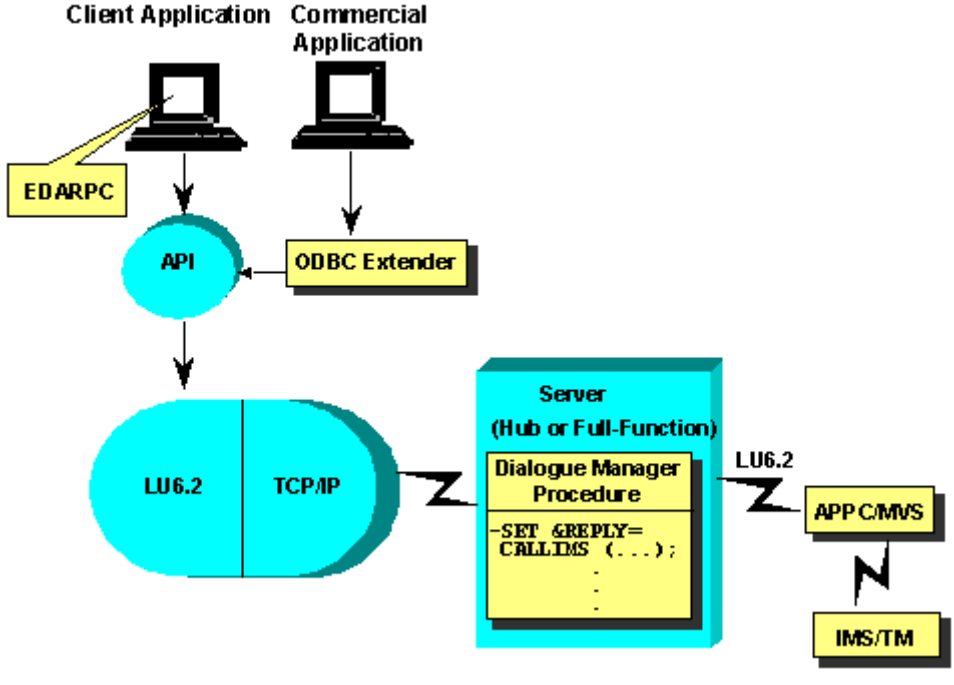
CALLIMS is executed from a Dialogue Manager procedure using the --SET control statement.

Parameters on the EDARPC call are sent by the client application using an EDARPC command and passed as a message to an IMS Transaction (MPP) Message Processing Program. The MPP performs the required work using IMS/TM facilities and returns messages to the application by inserting segments into the I/O Program Communication Block (PCB).

Taking advantage of APPC/IMS and APPC/MVS, CALLIMS provides implicit support for the execution of MPPs. This means that standard, existing MPPs (for example, the IMS IVP PART transaction) can interact with workstation front-ends or other client platforms. No changes are required in the MPPs.

Reference CALLIMS Processing in the Server Environment

The following figure illustrates CALLIMS processing in the server environment.



For a description of client components shown in the figure, see the *Introduction to iWay* manual. In the figure, a commercial application refers to third-party software (for example, Microsoft Visual Basic) that uses a client product that supports calls to stored procedures.

Overview: Transaction Server for IMS Configuration

Note: Before you begin configuration, verify that you have a valid CALLIMS procedure (see *CALLIMS Processing* on page 5-29). A sample is available in the *Stored Procedures Reference* manual. You may modify the sample or write your own.

To configure the Transaction Server for IMS/TM, you must follow these steps:

1. Ensure that you have required hardware and software.
2. Determine the APPC/MVS setup.
3. Set the security level.
4. Link-edit the API.
5. Confirm interaction With IMS/TM.
6. Verify installation of IMS/TM.

Step 1: Ensure That You Have Required Hardware and Software

The Transaction Server for IMS/TM requires the following hardware and software:

- An IBM or IBM-compatible mainframe running the MVS/ESA operating system.
- MVS/ESA Version 4, Release 2 or higher.
- APPC/MVS.
- ACF/VTAM Version 3.2, PUT Level 9001 or higher.
- IMS Version 4.1 or higher.

Step 2: Determine the APPC/MVS Setup

The Transaction Server for IMS/TM communicates with APPC/MVS to execute IMS/TM transactions. APPC/MVS includes an interface to IMS Release 4.1, called APPC/IMS. The APPC/IMS interface must be correctly configured to support IMS/TM transactions.

APPC/MVS components that may be used by the Transaction Server include:

- **APPLID for the IMS region.** This APPLID identifies applications executing IMS/TM transactions from LU6.2 devices. For examples of an LU6.2 APPLID that is used with APPC/IMS, see *Creating Network Definitions* in Chapter 2, *Preparing for Server Configuration*.
- **APPC/MVS side information data set.** This data set contains records keyed by a symbolic destination name. This is a client/server feature of APPC/MVS that allows a site to establish default information for an application wishing to execute IMS/TM transactions. Each record may contain default APPLIDs, log mode table names, and IMS transaction names.

- **TP profile data set.** This data set allows a site to substitute application-oriented transaction names with a real IMS transaction name. For example, assume that an application refers to a transaction named NAMECHG. The actual transaction (translated in the TP Profile Data set) may be SSNUPDT.

Note: The APPC/MVS Transaction Scheduler (ASCH) must be fully configured and running in order for the Transaction Server to communicate successfully with APPC/IMS. See the appropriate IBM APPC/MVS documentation for configuration instructions.

Step 3: Set the Security Level

To establish conversation security with APPC/IMS, define the environment under which the partner (the other end of the conversation with IMS/TM) will run.

The two possible security levels are:

- **Security Same.** Specifies that the transaction invoked by CALLIMS be run under the same security environment as the calling function (CALLIMS). The security environment includes the user ID and possibly the security profile name of the application.

  This security level is the default if no security information is explicitly provided in the CALLIMS function.

- **Security Program.** Specifies that the application provide a user ID, password, and security profile name to establish the security environment. This security level allows the partner to verify the information, and occurs when the information is provided to the Dialogue Manager procedure in which CALLIMS is invoked.

Step 4: Link-edit the API

The CALLIMS subroutine communicates with APPC/MVS using the APPC/MVS API. To establish communication between CALLIMS and APPC/MVS, you must link-edit the API.

To link-edit the API, submit the following job

qualif`.EDACTL.DATA(GENEAPPC)`

where:

qualif

Is the high-level qualifier for the data sets.

Step 5: Confirm Interaction With IMS/TM

The Transaction Server for IMS provides a sample REXX EXEC that helps confirm the successful installation of APPC/MVS and APPC/IMS. The EXEC will invoke an existing MPP.

Follow these steps to use the sample EXEC, which is located in *qualif*.EDACTL.DATA(APPCIVP).

1. The EXEC has default values to execute the IMS IVP PART transaction. You may need to change the following:

`TP_name = 'PART'`	The name of the transaction to invoke.
`Sym_dest_name = 'IMSDEST'`	The symbolic destination name found in the side information data set.
`Mode_name = 'LU62APPC'`	The log mode table entry name used by the LU6.2 APPLID supplied on the next keyword.
`Partner_LU_Name = 'SCMI41AX'`	The LU6.2 APPLID defined for use by APPC/IMS.
`Partnumber = 'AN960C10'`	The input data for the transaction.

2. In case of errors, APPCIVP calls standard REXX EXECs named IRXCKRC and ATBCKRC, so you must copy both EXECs from *qualif*.SYS1.SAMPLIB to *qualif*.EDACTL.DATA.

3. Once you have made the necessary changes to APPCIVP and copied IRXCKRC, run the EXEC from TSO as follows:

   ```
   EX 'qualif.EDACTL.DATA(APPCIVP)' EXEC
   ```

   This routine confirms that APPC/IMS is correctly configured to operate in a client/server environment. The next step enables the server to initiate an IMS connection.

Step 6: Verify Installation of IMS/TM

To verify that the installation of IMS/TM is correct, execute your CALLIMS Dialogue Manager procedure using the IVP on TSO. Instructions are located in Chapter 6, *Testing Server Communications*.

CHAPTER 6

Testing Server Communications

Topics:

- Modifying the Start-up JCL
- Loading Sample Files
- Running the Installation Verification Program (IVP)
- Next Steps

After you have generated your server and, optionally, configured server system features required by your site, you are ready to finish modifying the start-up JCL and submit the server job. Once the server has successfully started, you can verify your configuration.

Note: If you are configuring additional server products, such as FOCUS Database Server or Resource Analyzer and Resource Governor, you must modify the JCL again after you do so. It is recommended that you test your server both before and after configuring any additional products.

Modifying the Start-up JCL

The first step in testing your configuration is modifying the startup JCL file for your site. The member name depends on your server type. See Chapter 4, *Generating a Server* for more information about file names.

Procedure How to Modify the Start-up JCL

1. Edit *servertype*JCL as follows.

   Note: This example shows the JCL generated in member FFSJCL for a Full-Function Server. The *qualif* value used was EDAARH.V5R2M00.

```
//*************************************************************
//**  Sample JCL.                                            **
//*************************************************************
//*
//* Supply a proper job card.
//*
//EDASERVE  EXEC PGM=SSCTL
//STEPLIB    DD  DISP=SHR,DSN=EDAARH.V5R2M00.EDALIB.LOAD
//EDASERVE   DD  DISP=SHR,DSN=EDAARH.V5R2M00.INSTALL.DATA(FFSSERV)
//ERRORS     DD  DISP=SHR,DSN=EDAARH.V5R2M00.EDAMSG.DATA
//IBIFEX     DD  DDP=SHR,DSN=EDAARH.VSR2MOO.EDAPRC.DATA
//IBIMFD     DD  DISP=SHR,DSN=EDAARH.V5R2M00.EDAMFD.DATA
//FOCEXEC    DD  DISP=SHR,DSN=user.focexec.data
//*MASTER    DD  DISP=SHR,DSN=user.mfd.data
//*FOCPSB    DD  DISP=SHR,DSN=user.focpsb.data
//ACCESS     DD  DISP=SHR,DSN=EDAARH.V5R2M00.EDAAFD.DATA
//EDAPRINT   DD  SYSOUT=*
//IBISNAP    DD  SYSOUT=Q
//EDACSG     DD  DISP=SHR,DSN=EDAARH.V5R2M00.INSTALL.DATA(FFSCSG)
//*************************************************************
//**  Optional Console Configuration File.                   **
//*************************************************************
//*
//EDACONS    DD  DISP=SHR,DSN=EDAARH.V5R2M00.INSTALL.DATA(FFSCONS)
//*
//*************************************************************
//**  Optional ECC Restart File.                             **
//*************************************************************
//*
//*EDASTART DD DISP=SHR,DSN=EDAARH.V5R2M00.INSTALL.DATA(FFSJCL)
//*
```

2. Supply a valid JOB card for your site.

3. If you are using a pooled list of APPLIDs, you must add a DDNAME for the pooled list. For example:

   ```
   //LU6POOL     DD  DISP=SHR,DSN=EDAARH.V5R2M00.INSTALL.DATA(LU6POOL)
   ```

   If you are using a pooled list for LU6.2, you need to add a DDNAME for the list.

4. If you are configured to use Interlink TCP/IP, you must:

   a. Rename member qualif.EDALIB.LOAD(SASLIB) to SASLIBXX.

   b. Rename member qualif.EDALIB.LOAD(SASLIBTR) to SASLIB.

   c. Create a new PDS load library qualif.LSCNCOM.LINK (suggested name) with the following DCB:

   ```
   DBC=(LRECL=0,RECFM=U,BLKSIZE=6144),SPACE=(TRK,(10,5,1))
   ```

   d. Copy member LSCNCOM from your Interlink installation LINK library into the library created in the previous step (**c**).

   For Interlink TCP/IP **prior to Version 5.2**, the STEPLIB concatenation for the server JCL must be as follows:

   ```
   //STEPLIB     DD DISP=SHR,DSN=qualif.EDALIB.LOAD
   //            DD DISP=SHR,DSN=qualif.LSCNCOM.LINK
   //            DD DISP=SHR,DSN=interlink.LOAD
   ```

   For Interlink TCP/IP **Version 5.2**, the STEPLIB concatenation for the server JCL must be as follows:

   ```
   //STEPLIB     DD DISP=SHR,DSN=qualif.EDALIB.LOAD
   //            DD DISP=SHR,DSN=qualif.LSCNCOM.LINK
   //            DD DISP=SHR,DSN=interlink.LOAD
   //            DD DISP=SHR,DSN=qualif.EDASASC.LINKLIB
   ```

   Note: The above data sets can be combined into one library if the order of concatenation is maintained during the combination process. Before starting the server, see Chapter 8, *Modifying the Server Configuration*, for further steps regarding Interlink TCP/IP usage.

5. If you configured for IMS/DBCTL, uncomment and edit the following line with the DSN you created in *Customizing the IMS/DBCTL Environment* in Chapter 3, *Configuring Data Adapters*:

   ```
   //*FOCPSB      DD  DISP=SHR,DSN=user.focpsb.data
   ```

   Note: If you are using IMS Release 6.1, due to a change in requirements, the DFSRESLB allocation, if present, may cause DBCTL initialization problems. If this happens, remove DFSRESLB from your JCL and allocate the RESLIB.LOAD to the STEPLIB.

6. Submit the job.

Once you have successfully submitted this job, it should remain running, awaiting client connections. You are now ready to test the server's communications. If you are configuring a Hub or Full-Function Server, proceed to *Running the Installation Verification Program (IVP)* on page 6-4.

Loading Sample Files

Follow these instructions to execute the procedure TSCFOLD, which is used to load the sample files:

1. Edit *qualif*.EDACTL.DATA(TSCFOCLD). Its contents are shown below:

```
//LOADEM EXEC PGM=TSCOM3
//STEPLIB DD DISP=SHR,DSN=qualif.EDALIB.LOAD
//ERRORS  DD DISP=SHR,DSN=qualif.EDAMSG.DATA
//TRMIN   DD *
EX EDAFOCLD qualif
/*
//TRMOUT DD SYSOUT=*,DCB=(LRECL=133,BLKSIZE=133,RECFM=F)
//EDARPC  DD DISP=SHR,DSN=qualif.EDARPC.DATA
//EDAMFD  DD DISP=SHR,DSN=qualif.EDAMFD.DATA
```

2. Add a JOBCARD and change the text *qualif* to the high-level qualifier used for the server installation.

3. Submit the JCL. The procedure will create and load with data the following sample FOCUS files:

```
qualif.JOBFILE.FOCUS
qualif.EDUCFILE.FOCUS
qualif.EMPLOYEE.FOCUS
qualif.SALES.FOCUS
qualif.LEDGER.FOCUS
qualif.CAR.FOCUS
qualif.FINANCE.FOCUS
qualif.REGION.FOCUS
```

Running the Installation Verification Program (IVP)

The Installation Verification Program (IVP) tests the configuration by running a fixed set of tests against the server. The IVP tells you if a client and server on the same machine can talk to one another. If you want to create your own queries to test the server, use the RDAAPP client tool.

Once you have established that a client and server on the same platform can communicate, you have a successful foundation on which to configure clients on other platforms. It is therefore recommended that you run the IVP.

To run the IVP, you must configure the server for inbound communications and the client for outbound communications. See Chapter 4, *Generating a Server*, or Chapter 8, *Modifying the Server Configuration*, for more information. To make a successful connection, the following items must match in the client and server files:

- The client and server must be using the same protocol.
- The client node must match the service name in the MVS server.

Procedure **How to Run the IVP**

Complete the following steps to run the IVP provided with the server:

1. Edit the member *qualif*.EDARPC.DATA(IVPRPC8), and change *qualif* to the high-level qualifier of the installed libraries. There is only one reference to *qualif* in this RPC, on line 20. The member CCARS in *qualif*.EDAMFD.DATA contains the data used to load the test database CAR.

```
17 -IF &RETCODE NE 0 GOTO NODATA ;
18 -GOTO MAIN
19 -MVS1
20 DYNAM ALLOC FILE CCARS DA qualif.EDAMFD.DATA(CCARS) SHR REU
21 -* TSO DDNAME CCARS
```

2. To test a Full-Function Server, edit and run *qualif*.INSTALL.DATA(JRUNIVP). This job runs the server IVP. Complete the following steps:

a. Add a job card.

b. If you are using LU6.2 communications and have set up LU POOLs, uncomment the LU6POOL DD statement, and verify that the libraries these DD statements are pointing to are correct data sets and members.

c. If you are using Interlink TCP, the STEPLIB concatenation must match the servers and the clients communications configuration file, EDACS3. This file must be changed using the instructions for outbound blocks in Chapter 8, *Modifying the Server Configuration*.

d. The output of the run will be directed to SYSPRINT. SYSPRINT is, by default, directed to JES OUTPUT, CLASS=*. If you wish to keep the results, allocate a sequential file LRECL=133, SPACE=(CYL,(5,5)). Then change the SYSPRINT DD statement to point to this data set.

3. Submit the IVP JCL.

```
//* Job card goes here
//*************************************************************
//*                        E D A I V P T                      *
//*            (INSTALLATION & VERIFICATION TEST)             *
//*************************************************************
//* CALLING STRUCTURE.                                        *
//* ------------------                                        *
//*                                                           *
//*     EXEC PGM=EDAIVPT,PARM='PARM1'                         *
//* E.G EXEC PGM=EDAIVPT,PARM='EDAUSER'                       *
//*                                                           *
//*     PARM1 : SERVER NAME - REFER TO COMMS CONFIG           *
//*************************************************************
//RPCTEST   EXEC PGM=EDAIVPT,PARM='EDAUSER
//*************************************************************
//STEPLIB   DD DISP=SHR,DSN=EDAARH.V5R2M00.EDALIB.LOAD
//IVPIN     DD DISP=SHR,DSN=EDAARH.V5R2M00.EDARPC.DATA
//STDOUT    DD SYSOUT=*,DCB=LRECL=133
//EDACS3    DD DISP=SHR,DSN=EDAARH.V5R2M00.INSTALL.DATA(CS3CLNT)
//SYSPRINT  DD SYSOUT=*,DCB=LRECL=133
//SYSTERM   DD SYSOUT=*,DCB=LRECL=133
//SYSUDUMP  DD SYSOUT=*
//*IBITRACE  DD *
//* SET TRACEON=EDAAPI//STDOUT
//
```

Note: If you are using Interlink TCP/IP, you must make sure that the STEPLIB in the above JCL matches the Server JCL STEPLIB. For more information, see step 4 of *Modifying the Start-up JCL* on page 6-2.

4. The results of the IVP are directed to SYSPRINT. Review the results and look for any occurrences of the words “unsuccessful” or “error.” There is a sample output supplied in *qualif*.EDACTL.DATA(IVPOUT).

The IVP:

- Checks communications and displays the API version.
- Displays the list of server capabilities using EX EDASYS01.
- Checks the SQL translator SELECT.
- Checks all Metadata stored procedures to ensure that the catalog is fully functional:

  - ODBC RPC’s

  - TABLIST

- Checks the DMH Dialogue Manager:

  - Displays the server release.

  - Shows all the default server options.

  - Checks other basic functionality including &variables.

  - Calls the user-written subroutine GETUSER. Displays connected user ID.
- Checks TABH EDASQL non-relational access engine:

  - Builds the CAR file.

  - Reports from the CAR file.
- Verifies that CALLPGM is correctly installed.

Next Steps

After you have completed the installation and configuration process, the server is enabled for operations at your site. However, depending on your site requirements, you might need to optionally install or configure the following services:

- Resource Governor Governing Services.
- FOCUS Database Server.

For details, see Chapter 7, *Configuring Supported Services*.

If you are not installing or configuring any supported services or products, proceed to the *iWay Server Administration for MVS and VM* and *iWay Data Adapter Administration* manuals. These manuals assist you with data access and server customization tasks that need to be performed after the initial configuration:

- Setting user access to data sources and server functions.
- Customizing the server's global profile and user profiles.
- Creating Master Files and Access Files, which are required to retrieve user data.
- Working with the data adapters chosen for your site.

CHAPTER 7

Configuring Supported Services

Topics:

- FOCUS Database Server Support
- Resource Analyzer/Resource Governor Software Components
- Configuring Resource Analyzer and Resource Governor With Your Server
- Enabling Auto Procedures for an MVS Server
- Next Steps

Once the server is enabled for operations at your site, you may need to install or configure the following services, which are supported by the server:

- **The FOCUS Database Server.** This feature of FOCUS Services enables multiple users to write simultaneously to a FOCUS database without overwriting each other. See *FOCUS Database Server Support* on page 7-2.
- **Resource Analyzer and Resource Governor.** Sister products that provide usage monitoring and governing functions for requests processed by the server, and generate reports based on usage monitoring data. See *Configuring Resource Analyzer and Resource Governor With Your Server* on page 7-6.

FOCUS Database Server Support

The server for MVS supports the FOCUS Database Server (Simultaneous Usage) feature. This feature allows multiple users to write simultaneously to the FOCUS Database Server without overwriting each other.

Procedure **How to Configure the FOCUS Database Server**

1. Install the IBI subsystem. See Chapter 5, *Configuring Server System Features*, for instructions. The subsystem must be from this release; old subsystem code will not work.

2. Create the communications data set.

   The database job that performs the centralized FOCUS I/O runs in its own address space, and its presence in the system is made known through a communications data set that will be accessed by the adapter region. This data set plays a role only in the initial contact; the subsequent transfer of commands and data between the central database job and the server region takes place entirely in virtual storage.

   The communications data set must be allocated and catalogued on a permanently mounted volume, and its name must be chosen so as to allow WRITE access by the central database job and READ access by the server region. The allocation is for a minimal amount of space for a single16-byte fixed-length record. The actual data set name is irrelevant, but we recommend something suggestive of its role.

   You can use the following sample JCL to create the FOCSU data set:

   ```
   //* Job card goes here
   //*
   //STEP1     EXEC PGM=IEFBR14
   //FOCSU     DD   DSN=qualif.FOCSU.DATA,DISP=(NEW,CATLG),
   //          SPACE=(TRK,(1,1)),DCB=(LRECL=16,RECFM=F,BLKSIZE=16),
   //          UNIT=SYSDA
   /*
   ```

   It is possible that you will want to run several central database jobs, each dedicated to a set of FOCUS data sources associated with a particular application. To do so, you must allocate a different communications data set to each job.

3. Create an auxiliary database.

   This database is an auxiliary FOCUS data source that will be accessed in READ mode by the central database job, but not by the server region. The same auxiliary database can be used by all central database jobs if you have more than one database.

You can use the following sample JCL to create the FOCUSSU database:

```
//* Job card goes here
//*
//STEP1    EXEC PGM=FOCUS
//STEPLIB  DD   DSN=qualif.EDALIB.LOAD,DISP=SHR
//ERRORS   DD   DSN=qualif.EDAMSG.DATA,DISP=SHR
//MASTER   DD   DSN=qualif.EDAMFD.DATA,DISP=SHR
//FOCUSSU  DD   DSN=qualif.FOCUSSU.FOCUS,DISP=(NEW,CATLG),
//   SPACE=(TRK,(1,1)),DCB=(LRECL=4096,RECFM=FB,BLKSIZE=4096),
//   UNIT=SYSDA
//SYSPRINT DD     SYSOUT=*
//SYSIN    DD     *
 CREATE FILE FOCUSSU
 FIN
/*
```

The communications data sets, FOCSU and FOCUSSU, which were used with the IBI SVC, are still used with the IBI Subsystem, although the specifics of their usage differ.

The data sets are used to locate the address space that is to be communicated with, but are never actually opened.

Since the data sets are never actually opened, you may need to take steps to prevent HSM or similar products from migrating them off-line.

4. Install the security interface (optional).

The only installation step is the authorization of the EDALIB.LOAD library. You or your MVS systems programmer must authorize the entire EDALIB.LOAD library. This allows some of the modules in EDALIB.LOAD to issue certain privileged operating system functions, including security requests. The only module that actually issues these security requests is module SUSI. The HLISECUR module in EDALIB.LOAD is the only module with a non-zero authorization code, meaning that HLISECUR is the only module that can be invoked directly as an authorized program. All other modules in EDALIB.LOAD have an authorization code of zero, which means that they can run authorized only if they are invoked properly by HLISECUR.

The program HLISECUR links to the module HLISNK, which controls the sink machine. This program must run authorized so that the interface can later invoke the module SUSI to perform the security access checking.

The program SUSI checks security access on behalf of the source user. This program runs under the authorization of HLISECUR in order to issue the RACROUTE macro.

The modules HLISECUR and SUSI are the only modules in EDALIB.LOAD that must run authorized.

Source code for these modules is provided on the tape so that you are able to verify that these modules do not pose a system security risk when installed as authorized modules. The source code for HLISECUR and SUSI can be found in the EDALIB.DATA library in members HLISECUR and SUSISRC.

To use the interface, insert the following command in the FOCUS Database Server profile:

```
SET SUSI=ON
```

The FOCUS DATABASE SERVER profile is member HLIPROF of a PDS allocated to DDNAME FOCEXEC on the sink machine.

Once the interface is installed, you must change all sink machine jobs to execute the program HLISECUR, rather than HLISNK, as in the following JCL line:

```
//SINK  EXEC PGM=HLISECUR,PARM='parameters'
```

You can use the same password and ECHO or STAT parameters for HLISECUR as for HLISNK. For more information, see the *Simultaneous Usage Reference Manual, TSO Version*.

5. Create the FOCUS Database Server JCL.

The following is sample JCL for running the FOCUS Database Server:

```
//HLISNK   EXEC  PGM=HLISNK,REGION=960K
//STEPLIB  DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//*
//*        Communication data set
//*
//FOCSU    DD DSN=qualif.FOCSU.DATA,DISP=SHR
//*
//*        Auxiliary FOCUS database
//*
//FOCUSSU  DD DSN=qualif.FOCUSSU.FOCUS,DISP=SHR
//*
//          Sink error log
//*
//ERRORSDD DSN=qualif.EDAMSG.DATA,DISP=SHR
//HLIERROR DD SYSOUT=*,
//         DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//*
//*         Optional log file (can be a data set)
//*
//HLIPRINT  DD SYSOUT=*,
//          DCB=(RECFM=F,LRECL=88,BLKSIZE=88)
//*
//*         Descriptions of central databases
//*         Concatenated with the FOCCTL.DATA PDS
```

```
//*
//MASTER    DD DSN=qualif.EDAMFD.DATA,DISP=SHR,
//          DD DSN=user.edamfd.data,DISP=SHR,
//*
//*         ddnames of central FOCUS databases
//*         Use DISP=SHR for multi-threaded SU
//*
//ddname1   DD DSN=datasetname1,DISP=OLD
//ddname2   DD DSN=datasetname2,DISP=OLD
```

See the *Simultaneous Usage Reference Manual, TSO Version*, for optional parameters for this job.

6. Add user FOCUS data source ddnames to the JCL created in step 5.
7. Submit the JCL.

Resource Analyzer/Resource Governor Software Components

You must install and/or configure the following software components before you can begin using Resource Analyzer/Resource Governor:

- **Server.** A server must be installed and configured for Resource Analyzer/Resource Governor. The server houses the Resource Governor internal tables and processes data access requests. For more information, see the *iWay Server Installation* manual and Chapter 4, *Generating a Server*, in this manual. For information on the basic operations of Resource Analyzer/Resource Governor, see the *iWay Server Administration for MVS and VM* manual.
- **WebFOCUS.** To run Resource Analyzer and Resource Governor reports, you must select the options for Resource Analyzer and/or Resource Governor during the WebFOCUS client installation. For more information, see the *WebFOCUS and ReportCaster Installation and Configuration* manual for your platform.
- **Resource Analyzer and Resource Governor Web Application.** The Web application is required to run Resource Analyzer/Resource Governor reports. For details, see the *Data Management Administration Tools Suite Installation* manual.
- **Resource Analyzer/Governor Administrator.** Resource Analyzer and Resource Governor are used to control monitoring and build rules for data sources. To manage the administrative settings that control these activities, you must install the Resource Analyzer/Resource Governor Administrator. For information about using the Administrator, see the *Resource Analyzer Administrator's and User's Manual* and the *Resource Governor Administrator's and User's Manual*.

Configuring Resource Analyzer and Resource Governor With Your Server

Resource Analyzer allows you to generate reports based on usage monitoring data. Resource Governor provides usage monitoring and governing functions for requests processed by the server. Resource Analyzer and Resource Governor are available in a Full-Function or Hub Server environment.

You must create the Resource Analyzer/Resource Governor internal tables, ensuring that there is enough space for the usage monitoring process. The recommended space parameters for any data source depend on the query traffic through the Resource Analyzer/Resource Governor Usage Monitor. The amount of data recorded per request varies; a conservative estimate is 1,000 bytes of data per request. You should create the internal tables with this guideline in mind.

If you choose DB2 for the Resource Analyzer/Resource Governor internal tables, we recommend that you either create a segmented tablespace or allow DB2 to create its own tablespace for each table by specifying a database name only. This will assure faster lookup when retrieving records from the tables.

It is necessary to create the Resource Analyzer and Resource Governor internal tables before you can use Resource Analyzer and Resource Governor. For more information, see *How to Create Resource Analyzer/Resource Governor Internal Tables* on page 7-6.

Procedure How to Create Resource Analyzer/Resource Governor Internal Tables

1. Edit the server's global profile (EDASPROF) and turn Resource Analyzer/Resource Governor on by adding the following line:

   ```
   SET SMARTMODE=ON
   ```

2. Allocate and insert the *qualif*.SMARTLIB.DATA data set in the server startup JCL with the ddname SMARTLIB.

   Once Resource Analyzer/Resource Governor administrative tasks have begun, this data set will contain the following:

   - GKTABLE, a file that contains information about which data objects are being monitored and (for Resource Governor only) which data objects are being governed.
   - The rule file(s) used during the governing process (for Resource Governor only).

   For an illustration, see *Enabling Resource Analyzer and Resource Governor* on page 7-11.

3. If you are configuring Resource Governor, you must add an additional entry to the EDASERVE and EDACSG configuration sections of the JCL. The following example shows a sample EDACSG entry:

```
NODE          =LOOPBACK

BEGIN
  CLASS       =CLIENT
  PROTOCOL    =TCP
  HOST        =nnn.nn.nn.nnn
  SERVICE     =nnnn
    ; TRACE   =31
  COMPRESSION =1
END
```

where:

nnn.nn.nn.nnn

Is the IP address.

nnnn

Is the port number.

Note: Make sure the IP address and port number match the address and port of your MVS server. In addition, the node must be LOOPBACK.

The following shows a sample EDASERVE entry:

```
SERVICE       =LOOPBACK
PROGRAM       =TSCOM3
NUMBER_READY =1
MAXIMUM       =16
SERVINIT      =*,++
AUTOSTART     =NO
```

4. Allocate and insert the *qualif*.HTML.DATA data set in the server startup JCL with the ddname HTML. This data set stores HTML files used by Resource Analyzer and Resource Governor reporting.

5. Go to the Resource Analyzer/Resource Governor Administrator and connect to the appropriate server. You are prompted for the information that is required to create the Resource Analyzer and Resource Governor internal tables. See *Resource Analyzer/Resource Governor Administration Parameters* on page 7-9. The internal tables are created based on your entries. A message is displayed when this process is completed.

6. If you are using a *relational* data source as your Resource Analyzer/Resource Governor repository, the installation and configuration is complete.

If you are configuring Resource Governor, proceed to *Running the Resource Governor Configuration Verification Program* on page 7-14 before you use the Resource Governor Administrator to control monitoring of data sources and build rules.

If you are configuring Resource Analyzer only, you may begin to use the Resource Analyzer Administrator to control monitoring of data sources. For information about using the Administrator tool, see the *Resource Analyzer Administrator's and User's Manual*.

Or

7. If you are using *FOCUS* as your Resource Analyzer/Resource Governor database repository, you must activate the FOCUS Database Server (a feature of FOCUS that enables multiple users to read from and write to the same data sources concurrently).

To activate FOCUS Database Server, modify the *qualif*.EDACTL.DATA (GKESU) JCL and submit it. For a sample JCL, see *Using Resource Analyzer/Governor With FOCUS Database Server* on page 7-12.

After you submit the JCL, verify your installation and configuration before you continue with any administrative tasks. For details, see *Running the Resource Governor Configuration Verification Program* on page 7-14.

Procedure How to Disable Resource Analyzer/Resource Governor

If, at any time, you wish to disable Resource Governor/Resource Analyzer, you can edit the server profile, EDASPROF, and turn SMARTMODE off by commenting out the statement SET SMARTMODE=ON:

```
-*SET SMARTMODE=ON
```

Reference Resource Analyzer/Resource Governor Administration Parameters

You are prompted for the following information when you connect to the server from the Resource Analyzer/Resource Governor Administrator.

Required Information	Explanation
Owner Name	This value is required to use a relational database for collection. It serves as the owner name of the tables that are used for system administration and also, at run time, for collection by Resource Analyzer or Resource Governor. By default, the tables are created for the specified owner name in the default database; the owner must be authorized to create tables in this database. If you wish to create the tables in an alternate database where the owner is authorized to create tables, and if the technique is supported by the RDBMS, you can specify the database name as well (for example, database.owner). Consult the appropriate RDBMS documentation for the exact syntax. **Default:** None **Note:** The owner name is not applicable to FOCUS/FDS collection and may be left blank.
Collection Name	This value is used by Resource Analyzer or Resource Governor as the server name in collected data and on reports. Any 8-character identifier is acceptable, but the value should be unique across all servers where Resource Analyzer or Resource Governor is installed. **Default:** Host server name.

Required Information	Explanation
Collection Adapter	This value is used to identify which data adapter is used for storing Resource Analyzer or Resource Governor collection data, as well as the system administration tables. It also determines the SUFFIX used when creating the Master Files for these tables. **Default:** None
Server Type	This value is used to track the type of server being used to monitor Resource Analyzer or Resource Governor. **Default:** Full Function Server
High Level Qualifier	This required value is the high level qualifier used for the data sets that are created if FOCUS SU is selected as the repository. It is also used during the configuration process to load the sample custom rule file, GKECR, into SMPRL. GKECR is located in the EDACTL.DATA data set that is unloaded during the Server installation process. **Default:** None
FOCEXEC Dataset	This required value is used to identify the name of the EDARPC data set used to store the GKESET procedure that contains Resource Analyzer and Resource Governor settings. The complete data set name must be entered and must be in the concatenated list of data sets pointed to by the DDNAME FOCEXEC in the server JCL. You must also have write access to the data set. **Default:** None
Smartlib	This required value is used to create the SMARTLIB data set. This data set is used to store the Resource Governor Knowledge Base files and the GKTABLE Knowledge Base file that is used by Resource Analyzer and Resource Governor as a control file to determine what data sources are being monitored and governed. **Default:** None

Required Information	Explanation
DB Space	This value is used to identify the Tablespace or Database Space used when DB2 or DB2/VM have been selected as the repository for the collection and system tables. **Default:** The PUBLIC tablespace defined for DB2 or DB2/VM.

Reference

Enabling Resource Analyzer and Resource Governor

You must insert the *qualif*.SMARTLIB.DATA data set in the server startup JCL with the ddname SMARTLIB. Once Resource Analyzer/Resource Governor administrative tasks have begun, this data set will contain the following:

- GKTABLE, a file that contains information about which data objects are being monitored and (for Resource Governor only) which data objects are being governed.
- The rule file(s) used during the governing process (for Resource Governor only).

The following example illustrates where you may insert this library:

```
//EDA       EXEC PGM=SSCTL
//STEPLIB   DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//ERRORS    DD DSN=qualif.EDAMSG.DATA,DISP=SHR
//SMARTLIB  DD DSN=qualif.SMARTLIB.DATA,DISP=SHR
//FOCSU01   DD DSN=qualif.EDASU.DATA, DISP=SHR
```

See Chapter 6, *Testing Server Communications*, for server JCL samples.

Note: You must add the DD statement for FOCSU01 when you are using FOCUS as your Resource Analyzer/Resource Governor repository for usage monitoring data. For details, see *Using Resource Analyzer/Governor With FOCUS Database Server* on page 7-12.

Reference Using Resource Analyzer/Governor With FOCUS Database Server

If you are using FOCUS as your Resource Analyzer/Resource Governor database repository, you must activate the FOCUS Database Server by modifying the *qualif*.EDACTL.DATA(GKESU) JCL and submitting it before you can verify your installation and configuration, and before you can continue with any administrative tasks.

Edit the GKESU JCL as follows:

```
//*******************************************************************
//**         Sample Performance Management Suite SU JOB            **
//*******************************************************************
//*
//* Supply a proper job card.
//*
//HLISNK    EXEC PGM=HLISNK
//STEPLIB   DD DSN=qualif.EDALIB.LOAD,DISP=SHR
//FOCSU     DD DSN=qualif.EDASU.DATA,DISP=SHR
//FOCUSSU   DD DSN=qualif.FOCUSSU.FOCUS,DISP=SHR
//FOCSORT   DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//ERRORS    DD DSN=qualif.EDAMSG.DATA,DISP=SHR
//*HLIPRINT  DD SYSOUT=*,DCB=(LRECL=133,BLKSIZE=133,RECFM=F)
//MASTER    DD DSN=qualif.EDAMFD.DATA
//SMCNTRL   DD DSN=qualif.SMCNTRL.FOCUS,DISP=SHR
//SMPRMTRS  DD DSN=qualif.SMPRMTRS.FOCUS,DISP=SHR
//SMKBASE   DD DSN=qualif.SMKBASE.FOCUS,DISP=SHR
//SMPRL     DD DSN=qualif.SMPRL.FOCUS,DISP=SHR
//SMQUERY   DD DSN=qualif.SMQUERY.FOCUS,DISP=SHR
//SMREQSTS  DD DSN=qualif.SMREQSTS.FOCUS,DISP=SHR
//SMFROMS   DD DSN=qualif.SMFROMS.FOCUS,DISP=SHR
//SMCOLMNS  DD DSN=qualif.SMCOLMNS.FOCUS,DISP=SHR
//SMRELTNS  DD DSN=qualif.SMRELTNS.FOCUS,DISP=SHR
//SMFNCTNS  DD DSN=qualif.SMFNCTNS.FOCUS,DISP=SHR
//SMBYS     DD DSN=qualif.SMBYS.FOCUS,DISP=SHR
//SMGOVEND  DD DSN=qualif.SMGOVEND.FOCUS,DISP=SHR
//SMRPCS    DD DSN=qualif.SMRPCS.FOCUS,DISP=SHR
```

where:

qualif

Is the high-level qualifier for the data sets.

Submit this JCL after you have completed the Resource Analyzer/Resource Governor installation. The FOCUS Database Server job should be established as a high-priority, started task that is non-swappable. The FOCUS Database Server must be running in order for you to perform any of Resource Analyzer/Resource Governor's administrative tasks.

If you try to submit a job when the FOCUS Database Server is not running, you will receive an error message like the following in your job output:

```
ERROR AT OR NEAR LINE      7  IN PROCEDURE FOCSQL01FOCEXEC *
 (FOC542) SU. COMMUNICATION IS NOT AVAILABLE TO CENTRAL DATABASE MACHINE:
   BYPASSING TO END OF COMMAND
ERROR AT OR NEAR LINE      9  IN PROCEDURE FOCSQL01FOCEXEC *
 (FOC542) SU. COMMUNICATION IS NOT AVAILABLE TO CENTRAL DATABASE MACHINE:
 (FOC238) LINKED FIELD IS NOT INDEXED: SMCNTRL
 (FOC236) LINKED FILE DOES NOT HAVE A MATCHING KEY FIELD OR SEGMENT:
SMNAME
   BYPASSING TO END OF COMMAND
ERROR AT OR NEAR LINE      4  IN PROCEDURE FOCSQL02FOCEXEC *
 (FOC542) SU. COMMUNICATION IS NOT AVAILABLE TO CENTRAL DATABASE MACHINE:
   BYPASSING TO END OF COMMAND
```

Once you have submitted the GKESU JCL, you may run the Resource Governor Configuration Verification Program (CVP). See *Running the Resource Governor Configuration Verification Program* on page 7-14.

For more information, see the *FOCUS for IBM Mainframe User's Manual* and the *FOCUS for IBM Mainframe Simultaneous Usage Reference Manual*.

Granting Access to Resource Analyzer/Resource Governor Databases

Resource Analyzer and Resource Governor have two separate databases, one used for administration and the other for usage monitoring data. If you are using a relational database (that is, not FOCUS) as your Resource Analyzer/Resource Governor repository, a GRANT command is issued for all users (PUBLIC) when the internal tables are created. This command enables them to write to the usage monitoring database.

The owner ID specified when the internal tables are created is granted user-access rights to the administrative databases (SMCONTROL, SMKBASE, SMPRL, and SMPARAMETERS).

If you want to grant other users access to the administrative databases, you must issue additional GRANT commands for those user IDs. A sample data file, GKEGRANT, can be found in *qualif*.EDALIB.DATA. This file contains the commands needed to grant user-access to the Resource Analyzer and Resource Governor administrative databases. For sample syntax, see *How to Grant Access to an Administrative Database* on page 7-14.

Syntax How to Grant Access to an Administrative Database

The following syntax grants user-access to a Resource Analyzer/Resource Governor administrative database

```
GRANT SELECT, UPDATE, INSERT ON owner.admin_database TO user1, user2 ..
```

where:

owner

Is the owner name of tables that are used for system administration and collection.

admin_database

Is one of the Resource Analyzer/Resource Governor administrative databases. For a list, see *Granting Access to Resource Analyzer/Resource Governor Databases* on page 7-13.

user1, user2 ..

Are the user IDs of the users to whom you wish to grant access.

Note: To grant certain users access to the Resource Analyzer and Resource Governor administrative databases, you must issue the GRANT command for each database.

Running the Resource Governor Configuration Verification Program

The Configuration Verification Program (CVP) is applicable *only* to Resource Governor, *not* to Resource Analyzer.

The CVP is a batch program that verifies Resource Governor's installation and configuration. The following remote procedures (which are all located in *qualif*.EDARPC.DATA) are executed:

- GKECOL (executed from the script GKECOLL, which is located in *qualif*.EDACTL.DATA)
- GKEPARM (executed from the script GKEPARMS, which is located in *qualif*.EDACTL.DATA)
- GKEGOV (executed from the script GKEGOVN, which is located in *qualif*.EDACTL.DATA)
- GKERULE (executed from the script GKERULES, which is located in *qualif*.EDACTL.DATA)
- GKEDEL (executed from the script GKEDELE, which is located in *qualif*.EDACTL.DATA)

For more information on these procedures, see the *iWay Server Administration for MVS and VM* manual.

The CVP verifies the following functionality:

- Request usage monitoring (collection).
- Request governing based on collection.

The CVP uses a temporary table, GKEIVP, for validating usage monitoring and governing. For instructions, see *How to Run the Resource Governor Configuration Verification Program* on page 7-15.

Procedure How to Run the Resource Governor Configuration Verification Program

To run the CVP, perform the following:

1. Edit the data set, *qualif*.EDACTL.DATA(GKEIVP) to modify the file allocations for site specific high-level qualifiers. Make sure the TRMOUT allocation is satisfactory. As delivered, this file content goes to the JES output queue. The following are the lines that will need editing:

```
//STEPLIB  DD  DSN=qualif.EDALIB.LOAD,DISP=SHR
//SMARTLIB DD  DSN=qualif.SMARTLIB.DATA,DISP=SHR
//ERRORS   DD  DSN=qualif.EDAMSG.DATA,DISP=SHR
//MASTER   DD  DSN=qualif.EDAMFD.DATA,DISP=SHR
//ACCESS   DD  DSN=qualif.EDAAFD.DATA,DISP=SHR
//FOCEXEC  DD  DSN=qualif.EDARPC.DATA,DISP=SHR
```

2. Submit the JCL.

   Examine the TRMOUT file located in the JES output queue. If the Resource Governor installation and configuration are successful, the following messages are included among the output:

   - This statement indicates that the test data has been put into the test database:

```
*********************************************
*  INSERTS COMPLETED FOR TEST DATA          *
*********************************************
```

   - This statement indicates that the Resource Governor utility procedure, GKECOL, has successfully completed, and that data about test requests will be temporarily logged in the Usage Monitor.

```
*********************************************
*  GKECOL COMPLETED SUCCESSFULLY            *
*********************************************
```

   - This statement indicates that the test requests using SELECT against the GKEIVP test file have completed successfully, and that usage monitoring data has been populated in the Resource Governor usage monitoring database.

```
*********************************************
*  REQUESTS COMPLETED FOR TEST COLLECTION   *
*********************************************
```

- This is a cancellation message that Resource Governor generates to indicate that a SELECT statement was issued and cancelled based on rules built by the CVP.

```
*********************************************************
*  THIS QUERY SHOULD BE CANCELED BY RESOURCE GOVERNOR     *
*********************************************************
(GKE36048)  RESOURCE GOVERNOR CANCELED THIS REQUEST.   Governing
Mode = GOVERN
KBName = IVP   RuleNum = 3 Threshold Type = ROWS Thresh Exceeded =
10
```

If the Resource Governor installation and configuration is unsuccessful, error messages (preceded by the keyword ERROR) are displayed among the output, which are specific to the type of error encountered. For example:

- The following error message results from the GKECOL procedure not executing correctly:

  ```
  ERROR SETTING COLLECTION ON
  ```

- The following error message results from the GKETABLE procedure not executing correctly:

  ```
  ERROR CREATING GKTABLE
  ```

Tip: If the success messages are not found in the TRMOUT listing, or if error messages are found, turn tracing on in the CVP JCL. Re-submit the CVP and examine the messages in the trace for any problems in setup.

Using Resource Analyzer/Resource Governor Trace Files

Resource Governor/Resource Analyzer's internal trace component is called GKE.

For more information on how to turn on individual tracing systems, what is listed in a trace, and how to determine what is going on in the server when a request is processed, see the *iWay Server Administration for MVS and VM* manual.

Enabling Auto Procedures for an MVS Server

You can create Master and Access Files from a native file schema using two methods:

- Auto procedures
- Cobol FD Translator

To access an existing table or view using the server, you must first describe it to the server in two files: a Master File and an Access File. The Auto Tools use existing catalog definitions of tables and views to automatically generate these Master and Access Files.

Note: If you are using any of the Auto Tools to create Master and Access Files, you must allocate the load libraries for the database engine to the EDAAUTO environment. For example:

DB2IBM	DB2 Load Library
IDMS	Computer Associates IDMS Load Library
ADABAS	Software A.G. ADABAS Load Library
DATACOM	Computer Associates DATACOM Load Library

Since the Auto Tools are executed through the REXX EXEC, EDAAUTO, in *qualif*.EDACTL.DATA, you can use one of the following methods:

- Include the load libraries in the MVS LINKLIST.
- Allocate the load libraries in the STEPLIB DDNAME of the TSO logon procedure.

Note: The default access for IDMS is for release 12 or higher. If you require release 10 support, copy *qualif*.EDAMFD.DATA(IDMSI10M) to *qualif*.EDAMFD.DATA(IDMSIDD).

Procedure How to Install the EDAAUTO Facility

To install the auto procedures, perform the following:

1. Edit the clist *qualif*.EDACTL.DATA(EDAAUTO)

where:

qualif

Is the high-level qualifier for EDACTL.DATA.

2. Change *qualif* in the EDAAUTO clist to the high-level qualifier of the data sets.

```
ALLOC F(ERRORS)    DA('qualif.EDAMSG.DATA')          SHR REU
ALLOC F(EDAMFD)    DA('qualif.EDAMFD.DATA')          SHR REU
ALLOC F(EDAAFD)    DA('qualif.EDAAFD.DATA')          SHR REU
ALLOC F(EDARPC)    DA('qualif.EDARPC.DATA')          SHR REU
ALLOC F(PROFILE)   DA('qualif.EDARPC.DATA(EDAAUTO)') SHR REU

CALL 'qualif.EDALIB.LOAD(EDASAF)'
```

3. Run the REXX EXEC *qualif*.EDACTL.DATA(EDAAUTO).

The main menu for execution of the auto procedures appears:

```
Instructions : Place cursor on the type of File
               Description you wish to translate
               and then press ENTER.
               Use PF3 to Quit.

                        DB2

                        IDMS

                        ADABAS

                        DATACOM

F1= Help   2=      3= Quit    4=      5=     6=
```

Procedure How to Install the Cobol FD Translator

Important: The Cobol FD Translator is provided on a separate tape.

The Cobol FD Translator automatically creates data source descriptions from existing Cobol File Definitions (FDs). Perform the following steps:

1. Edit the clist

   qualif.EDACTL.DATA(EDACTF)

   where:

   qualif

   Is the high-level qualifier for EDACTL.DATA.

2. Change *qualif* within the EDACTF clist to the high-level qualifier of the data sets.

   ```
   SET &CTFLOAD=qualif.CTFR2.LOAD
   SET &CTFFOCEXEC=qualif.CTFR2.FOCEXEC.DATA
   SET &PRODLOAD=qualif.EDALIB.LOAD
   SET &PRODERRORS=qualif.EDAMSG.DATA
   ```

3. Run the clist *qualif*.EDACTL.DATA(EDACTF).

The Cobol FD Translator main screen appears:

```
            Cobol FD Translator - Version 2.0
  ENTER or OVERRIDE the following REQUIRED Parameters to continue

     COBOL FD   ==>
                (File containing COBOL record layout)

     Function   ==>  ADD
                (ADD to create new Master, REPL to replace Master)
     Master ==>  EDAMFD.DATA
                (Name of Master Library)
     Member ==>
                (Name of Member in Master Library)


  PF01 Help    PF03 Terminate                      ENTER Continue
```

See the *Cobol FD Translator* manual for instructions on how to proceed.

Next Steps

After configuring any supported services required by your site, proceed to Chapter 6, *Testing Server Communications*, to submit the server JCL and test your server with the supported services.

If you need to make any changes to your configuration, proceed to Chapter 8, *Modifying the Server Configuration*. Otherwise, proceed to the *iWay Server Administration for MVS and VM* manual to customize your server environment.

CHAPTER 8

Modifying the Server Configuration

Topics:

- EDACFG Configuration Files
- Changing Your Server Configuration
- Setting Server Security
- Enabling Usage Accounting
- Configuring the Server Console
- Configuring IMS/DBCTL Access
- Configuring Services for Inbound Protocols
- Modifying the Communications Subsystem Configuration File
- Configuring Inbound Communications
- Configuring Outbound Communications

You can manually modify the files generated by the EDACFGx procedures during installation in order to:

- Set server security.
- Enable usage accounting.
- Configure the server console.
- Configure access to the IMS/DBCTL environment.
- Modify the Communications Subsystem Configuration file.
- Configure inbound communications.
- Configure outbound communications.

For additional customization of your server, see the *iWay Server Administration for MVS and VM* manual.

EDACFG Configuration Files

The following files are generated during installation by the EDACFGx procedures in *qualif*.INSTALL.DATA:

	Member Name in *qualif*.INSTALL.DATA	
Reference Name	**Full-Function Server**	**Hub Server**
*servertype*CONS	FFSCONS	HUBCONS
*servertype*CSG	FFSCSG	HUBCSG
*servertype*JCL	FFSJCL	HUBJCL
*servertype*SERV	FFSSERV	HUBSERV

where:

*servertype*CONS
: Is the server console communications configuration file.

*servertype*CSG
: Is the server communications configuration file.

*servertype*JCL
: Is the server JCL file.

*servertype*SERV
: Is the server configuration file.

If you did not use the EDACFGx procedures to configure your server, we recommend that you create the same member names in *qualif*.INSTALL.DATA before you continue to modify your server configuration. See *How to Create Member Names in qualif.INSTALL.DATA* on page 8-3.

Procedure How to Create Member Names in qualif.INSTALL.DATA

If you do not already have the files created by the EDACFGx procedures and wish to modify aspects of your server configuration, edit the member names as follows:

1. If you are configuring the server console, create member *servertype*CONS and copy *qualif*.EDACTL.DATA(EDACONS) into the member.
2. Create member *servertype*CSG and copy *qualif*.EDACTL.DATA(EDACSG) into the member.
3. Create member *servertype*JCL and copy *qualif*.EDACTL.DATA(JCLSRV) into the member. You must edit this member and change *qualif* to the high-level qualifier that you used to install the server for MVS. Change all *servertype* values to the server type that you are manually configuring. For server types RDB2 and RDBC, delete the MASTER and ACCESS ddname statements. Otherwise, uncomment these two statements and supply a valid user Master data set name.
4. Create member *servertype*SERV and copy *qualif*.EDACTL.DATA(EDASERVE) into the member. This will contain global keywords and commented out server console keywords.
5. If you configured for IMS DBCTL access, append *qualif*.EDACTL.DATA(IMSDBCTL) to *servertype*SERV.

Changing Your Server Configuration

Changing your server configuration is a manual process. You open the configuration files, add or change the parameters, save the file, and recycle the server.

It is not necessary to rerun the EDACFGx procedure. If you do rerun the EDACFGx procedure using the same high-level qualifier for your data sets, you erase your previous configuration, including all profiles and site-specific information.

Note: The sample files used in illustrations are for a Full-Function Server. Also, any reference to a Hub Server includes a Full-Function Server with Hub Services.

Reference Comment Lines in the Configuration Files

In the communications files that are copied to EDACSG, comments are preceded by a semicolon (;). In all other files (for example, the EDASERVE file in *servertype*SERV), an asterisk (*) precedes comments.

When you are instructed to uncomment a line in a file, delete the applicable character (semicolon or asterisk).

Setting Server Security

Security is set to off when you generate the EDASERVE file in *servertype*SERV. The first time you bring up the server to test communications (for example, when you run the IVP), it is recommended that you leave security disabled. Once you know that the user can connect to the server, reset the value of EXTSEC to enable security, typically those privileges associated with the submitter's user privileges.

The function of server security is to enforce access to the server on a per-user basis. Server authentication processing ensures that clients attempting to access the server are authenticated during connection processing. The server manages authentication by interfacing with MVS Security using the System Authorization Facility (SAF). In addition, the SAF Interface allows the server to use a common interface to the available security packages, such as CA-ACF2, CA-TOP SECRET, and RACF, thus respecting all authentication and access rules applied through these packages.

Using SAF and External Security Packages

The server uses SAF in conjunction with security packages, such as CA-ACF2, RACF, and CA-TOP SECRET, to identify and authenticate remote users and to control their access to files and data sources. Each user is therefore subject to the security restrictions of the server platform.

To enable external security packages to operate in this fashion, you must:

- Include the EXTSEC parameter in your server configuration file and have it set to ON. (By default, EXTSEC is included and set to the default value of OFF.) When EXTSEC=ON, you obtain the security restrictions defined by the external package for that user ID. When EXTSEC=OFF, you obtain the security restrictions that apply to the server job or started task.
- Ensure that *qualif*.EDALIB.LOAD is APF-authorized.

Before your first request, your client application must send either a trusted user ID, or a user ID and password (known to the MVS Security Subsystem), to the server in a connect request.

When EXTSEC is ON during connection processing, the necessary SAF calls are made to authenticate the user ID. The connection request fails if the Security Manager denies the user access. If authenticated, the connection is accepted and an agent is started and assigned the privileges of the authenticated user ID. The external security manager then controls access to resources. For more information on authentication, refer to the *iWay Server Administration for MVS and VM* manual.

If the server is a started task, you must ensure that it has no special security authorization, as is common with started tasks. If the server has special security authorization, it will be unable to enforce user-level security.

Note: Under IBM's RACF, a Started Task table allows the site to specify that the server for MVS will not have special authorization. Other security packages have different requirements.

Server Security Interfaces

The server creates a security environment for the client during the connection process. While the server supports any security package that supports the IBM SAF Interface, RACF security is the default; it is installed with the operating system. Therefore, the product certification is done under RACF.

Other packages such as CA-ACF2, Guardian, and CA-TOP SECRET are SAF-compliant and should work as documented. Call your local support center if you experience any security-related difficulties.

If you are implementing security for the default RACF package, each user must be defined to RACF and given access to the appropriate system resources.

For details about implementing user-level security for the server with external packages, see *How to Implement Security for CA-ACF2* on page 8-6 and *How to Implement Security for CA-TOP SECRET* on page 8-7.

How Server Security Works

The server provides user-level security using IBM's SAF. The server issues an SAF macro, RACROUTE, which creates an ACEE control block for each user task when a connection to the server is established. The supplied user ID is inserted into the ACEE, and it is this user ID that dictates which resources the server task can access. After the ACEE is created, whenever the server task attempts to access system resources, the MVS System Authorization Facility will issue a call to verify the user ID's authority to access the resource.

In order for user-level security to operate successfully, the following must be true for your site:

- The server STEPLIB allocation must be APF-authorized.
- The operating system security package must be set up to support MVS System Authorization Facility calls in order to process them correctly. Also, depending on the security package, the address space level (JOBNAME) security must have either no access to any system resources or full access to all resources. Refer to the security package's documentation to determine what level of access to the server address space is necessary to enable user-level security.

  Note: Every user ID used to access the server must be defined to the security package and given access to the appropriate system resources.

Procedure How to Implement Security for CA-ACF2

This section provides instructions on implementing user-level security with Computer Associates CA-ACF2 (Release 5.2 or higher). Refer to the security vendor's documentation for additional details, including available options.

To use CA-ACF2 for user-level security, perform the following steps:

1. You must alter the CA-ACF2 GLOBAL SYSTEM OPTIONS (GSO OPTS) records to enable the MVS System Authorization Facility.

   To identify the server program as a user of the MVS System Authorization Facility services, you must add three SAFPROT (SAF protection) records. Use the following syntax to define these records:

   ```
   classes (verify,dataset) cntlpt(SSFOC) subsys(SSFOC)
   classes (dataset) cntlpt(SSFOC) subsys(SVC019)
   classes (-) cntlpt(SSCON) subsys(SSCON)
   ```

   Note:

   - In order to verify APPL and PROGRAM SAF calls, the first SAFPROT record class should be changed to a dash (-). In this case, CA-ACF2 rules must enable individual users to access APPL and PROGRAM entities.
   - For all CA-ACF2 sites using Release 6.0 or higher, the logon ID record remains the same but SAFPROT records are no longer needed. This is because Release 6.0 directly supports RACROUTE calls, while in Release 5.x, RAC calls were not supported unless defined by SAFPROT.

2. The MVS address space must have access to those system resources that are required by each user. CA-ACF2 will check for job-level access as well as user-level access. Therefore, the job-level user ID must have access to all data sets. For example, this can be done by setting the MAINT attribute on the CA-ACF2 record for the job-level user ID. Refer to CA-ACF2 technical reference guides for further information.

3. The job-level user ID of the server should have the Multiple User, Single Address Space (MUSSAS) attribute set on. If the server is run as a started task, you must enable the started task attribute for the job-level user ID.

4. Each user ID employed must be defined to CA-ACF2.

Note: Any questions or problems regarding the functionality of CA-ACF2 security should be directed to Computer Associates CA-ACF2 Technical Services.

Procedure How to Implement Security for CA-TOP SECRET

If you use Computer Associates CA-TOP SECRET, follow the guidelines described here or refer to the security vendor's manual for implementing its SAF Interface for user-level security.

CA-TOP SECRET maintains security characteristics for each user and program in a "facility entry." Through a facility entry, the server must be defined to CA-TOP SECRET as a multi-user program requiring task-level authority.

To use CA-TOP SECRET, perform the following steps:

1. Create a CA-TOP SECRET facility entry for the server security module, SSFOC.

   Note: Do not use the facility entries for TONE or VAMSPF. Any other facility entry is acceptable.

2. Within this entry, include CA-TOP SECRET parameters to establish the proper operating characteristics.

3. Each user of the server must be defined to CA-TOP SECRET and given access to the appropriate system resources.

Example Defining the Server to CA-TOP SECRET

The following is an example of a facility entry that defines the server to CA-TOP SECRET:

```
FACILITY DISPLAY
INITPGM=SSFOC       ID=9 TYPE=26

ATTRIBUTES=IN-USE,ACTIVE,SHRPRF,ASUBM,TENV,NOABEND,MULTIUSER,NOXDEF
ATTRIBUTES=LUMSG,STMSG,SIGN(M),NOPSEUDO,INSTDATA,NORNDPW,AUTHINIT

ATTRIBUTES=NOPROMPT,MENU,NOAUDIT,RES,NOMRO,WARNPW,NOTSOC
ATTRIBUTES=NOTRACE,NOLAB,NODORMPW,NONPWR,NOIMSXTND
MODE=IMPL
LOGGING=ACCESS,INIT,SMF,MSG,SEC9
UIDACID=8 LOCKTIME=000 DEFACID=*NONE*   KEY=8
```

The underscored parameters are those necessary for server user-level security and are defined below. The other values can be set according to the needs of your site.

`INITPGM=SSFOC`	Program name to be registered with CA-TOP SECRET.
`TENV`	Specifies task level authority.
`MULTIUSER`	Specifies multi-user access.

Note: The newer versions of CA-TOP SECRET do not require the TENV or MULTIUSER parameters. Check the version installed at your site to verify whether you need to include these parameters.

Enabling Usage Accounting

The server uses the System Management Facility (SMF) for gathering and recording usage information. To enable the server to use this feature, the server program library, *qualif*.EDALIB.LOAD, and all additional load libraries in the server STEPLIB allocation must be defined to MVS as APF-authorized libraries.

You activate usage accounting by uncommenting the SMFNUM parameter in the EDASERVE file *servertype*SERV. The default value is 0. This value enables internal usage accounting for operator command displays and for the server console without generating the SMF records.

See the *iWay Server Administration for MVS and VM* manual for more information about this parameter.

Configuring the Server Console

Tip: If you have not configured the server console using the EDACFGx procedure and you do *not* want to use the console, skip this section.

The server console has two components:

- EDACONS ddname
- Security setting in the EDASERVE ddname

The EDACONS ddname, which is in the server JCL, *servertype*JCL, points to the console communications file. If you configured the console in the EDACFGx procedures, this file already contains the correct values. If you did not configure the console but now wish to do so, see *How to Configure the Server Console Communications File (EDACONS)* on page 8-9.

Procedure How to Configure the Server Console Communications File (EDACONS)

1. Edit the new member in *qualif*.INSTALL.DATA(*servertype*CONS) and copy *qualif*.EDACTL.DATA(EDACONS).

   Change the *lu2_applid* to the value of the LU2 applid that you have defined for use by the Console. For example:

```
*-------------------------------------------------------*
*       Sample Console Configuration File.              *
*   The only thing that should be changed is 'lu2_applid' *
*-------------------------------------------------------*
entity_name               = MVS_APPL
hermes_api_trace          = 0
service                   = EDACONS
entity                    = MVS_APPL
protocol                  = LU2
simulate_full_duplex      = OFF
retain_connection         = OFF
local_lu_name             = lu2_applid
lu2_trace                 = 0
```

   Note: The console uses a single LU2 APPLID to monitor all protocols.

2. Edit *qualif*.INSTALL.DATA(*servertype*JCL) and uncomment the following line:

```
//*EDACONS DD DDSP=SHR,DSN=qualif.INSTALL.DATA(servertypeCONS).
```

   The *qualif* should have already been changed to the high-level qualifier that you used to install the server software. For more information, see Chapter 2, *Preparing for Server Configuration*.

Procedure How to Configure Server Console Security Settings

Security settings for the server console are in the EDASERVE file *servertype*SERV.

If you activated the console using the EDACFGx procedure, the default security profile is already set up. The default security profile is CONSEC=INTERNAL. The other console parameters use the OWNERID parameter that you specified during the EDACFGx procedure. For more information, see *Setting Server Console Security* on page 8-10.

Setting Server Console Security

The server console offers two modes of controlling access: internal and external security. These two modes are defined by the CONSEC parameter in the EDASERVE file in the *qualif*.INSTALL.DATA configuration file.

- If INTERNAL security is enabled, then the server will use access lists provided in the server configuration file to control console access. See *Setting Internal Security for the Server Console* on page 8-10 for more information.
- If EXTERNAL security is enabled, the server invokes the System Authorization Facility (SAF) and the installed security subsystem (RACF, CA-ACF2, CA-TOP SECRET) to verify the user ID attempting to gain access to the Console. For more information, see *Setting External Security for the Server Console* on page 8-12.

The server console provides three levels of security:

- **Operator authority.** Enables a user to issue server operator commands, which affect the region's operation. This includes actual server operator commands, as well as operator-style commands issued from the panels.
- **Detail authority.** Enables a user to view details about other users, such as their current remote procedure and which files are allocated to the user.
- **Logon authority.** Enables a user to log on to the Console. Logon authority enables a user to see I/O and CPU statistics and to see which user IDs and services are active. Users with logon authority normally have Detail and Operator authority over users who are logged on the server with the ID identical to the one that was used to log on to the console.

Setting Internal Security for the Server Console

When console security is INTERNAL, the server's security interface can be either active or inactive. This is accomplished with the EXTSEC parameter in the server configuration file:

- If External Security is off (EXTSEC=OFF), the server uses the user ID list to authenticate the connection to the console.
- If External Security is on (EXTSEC=ON), the server sends the user ID and password to SAF and the external security package for authentication.

For more information on external security, see *Setting Server Security* on page 8-4.

Syntax

How to Set Server Console Security to Internal

Internal is the default setting for server console security. To set this parameter, uncomment and specify the following parameters in the EDASERVE file located in *servertype*SERV.

```
*****************************************************************
*   Full Function Data Adapter
*   Global Configuration File
*   Generated on 2 Oct 2001 at 16:37:55
*   The lowercase parameters on the right of the equals sign
*   are site specific and need to be modified during the
*   configuration of the Data Adapter.
EXTSEC      = OFF
*SMFNUM     = 0
FASTLOAD    = **AUTO**
LONGSYNM    = ON
LICENSE     = 100-111-0422
****************   Optional Console Parameters   ****************
*
*CONSEC     = INTERNAL
*CONSOPER   = userid1,userid2
*CONSDTL    = userid1
*CONSUSER   = *
*
****************   End of Console Parameters   *****************
```

where:

`INTERNAL`

Enables console authentication.

`userid1,userid2...`

Specifies users who have Operator privileges.

`userid3`

Specifies users who have Detail privileges (usually the same users specified for CONSOPER).

`*`

Specifies users who can log on to the console.

Note the following conventions when making the user ID declarations:

- '?' represents one wild card character in the position where it is specified in the user ID string.
- '*' represents any number of characters and may appear anywhere in the user ID string. When a user ID is specified in the CONSOPER list (to enable the server Operator privilege) or the CONSDTL list (to enable the operator to see detailed information), logon rights are automatically provided to the console. However, Operator or Detail rights are granted separately, and a user ID must be placed in both lists if both types of privileges are required.
- To specify more than one user ID for a given list, separate each user ID with a comma (,). If it is necessary to continue to another line, repeat the parameter and continue with the user ID list. For example, if the user IDs A, B, C, D, E, and F could not fit on a single line, the resulting security definition would appear as follows:

```
CONSUSER=A,B,C
CONSUSER=D,E,F
```

Setting External Security for the Server Console

Through the use of IBM's System Authorization Facility (SAF), you can verify the authorization of a resource, such as a data set, by naming two attributes of that resource: 'class' and 'entity'. In addition, the level of access must be indicated as READ (READ is the only access level checked for) and the ID of the user attempting the access.

The entity names look like data set names for convenience. You can change the class and entity names by assembling and linking new names.

Since the rules are checked using SAF, all users must be defined as having access to the logon rule. If the user does not have access to the Logon/User rule, then the Operator and Detail rules are not checked. For related information, see *How to Change Default External Rules* on page 8-13.

When external security is used, CONSEC=EXTERNAL must be set and the server's security must be active (EXTSEC=ON). After a valid user ID and password are accepted by the server console, three tests are made using SAF calls:

- READ access to the Logon/User rule.
- READ access to the Operator rule.
- READ access to the Detail rule.

All of the rules must share the same security class definition. The default class is FACILITY and the default entity names are:

- IBI.CONSOLE.LOGON for Logon/User access.
- IBI.CONSOLE.OPERATOR for Operator access.
- IBI.CONSOLE.DETAIL for Detail access.

These defaults are specified in *qualif*.EDALIB.DATA(SSENTTAB).

For specific implementations, see *Using RACF Security With the Server Console* on page 8-13, *Using CA-TOP SECRET Security With the Server Console* on page 8-14, and *Using CA-ACF2 Security With the Server Console* on page 8-14.

Procedure How to Change Default External Rules

You change default external rules by updating SSENTTAB (with new class and entry values), assembling and linking into *qualif*.EDALIB.LOAD, or specifying class using the CONSCLASS keyword and Operator, Detail, and Logon resources using the CONSENTO, CONSENTD, and CONSENTL keywords in the service configuration file. For example:

```
CONSCLASS      =      EDACONS
CONSENTO       =      EDA.CONSOLE.OPERATOR
CONSENTD       =      EDA.CONSOLE.DETAIL
CONSENTL       =      EDA.CONSOLE.LOGON
```

This example shows a resource class of EDACONS with the resource entries specified for Logon, Detail, and Operator access.

Example Using RACF Security With the Server Console

When RACF security is used, CONSEC=EXTERNAL must be set and the server's security must be active (EXTSEC=ON).

Below is an example of the RACF rules that apply to the definitions of external security, as specified in the server configuration file.

```
RDEFINE facility ibi.console.logon uacc(read)
RDEFINE facility ibi.console.detail uacc(none)
RDEFINE facility ibi.console.operator uacc(none)
PERMIT  ibi.console.operator class(facility) uacc(read) id(sys user1)
PERMIT  ibi.console.detail class(facility) uacc(read) id(sysmgr user2)
```

Example Using CA-TOP SECRET Security With the Server Console

When CA-TOP SECRET security is used, CONSEC=EXTERNAL must be set and the server's security must be active (EXTSEC=ON).

Below is an example of a CA-TOP SECRET definition for the Console and the user of this facility:

```
FACILITY DISPLAY FOR EDACON
INITPGM=SSCONS   ID=F  TYPE=99
ATTRIBUTES=IN-USE,ACTIVE,SHRPRF,ASUBM,NOABEND,MULTIUSER,NOXDEF
ATTRIBUTES=LUMSG,NOSTMSG,SIGN(S),INSTDATA,RNDPW,AUTHINIT
ATTRIBUTES=NOPROMPT,NOAUDIT,RES,WARNPW,NOTSOC,LCFTRANS
ATTRIBUTES=MSGLC,TRACE,NOEODINIT,NODORMPW,NOPWR,NOIMSXTND
MODE=WARN  DOWN=GLOBAL  LOGGING=INIT,SMF,MSG
UIDACID=8 LOCKTIME=000 DEFACID=*NONE*   KEY=8
MAXUSER=03000  PRFT=003
FACILITY(USER1=NAME=EDACON)
FACILITY(EDACON=PGM=SSCONS,ID=F)
FACILITY(EDACON=ACTIVE,SHRPRF,ASUBM,NOABEND,MULTIUSER,NOXDEF)
FACILITY(EDACON=LUMSG,NOSTMSG,SIGN(S),NOPSUEDO,
INSTDATA,NORDNPW,AUTHINIT)
FACILITY(EDACON=AUTHINIT,NOPROMPT,MENU,NOAUDIT,RES,MOMRO,
WARNPW,NOTSOC)
FACILITY(EDACON=TRACE,NOLAB,NODORMPW,NONPWRM,NOIMSXTND)
FACILITY(EDACON=MODE=WARN,LOG=ACCESS,LOCKTIME=000)
```

Note: The TENV parameter is not required for CA-TOP SECRET Release 4.3 or higher, but is required for previous releases.

Example Using CA-ACF2 Security With the Server Console

When CA-ACF2 security is used, CONSEC=EXTERNAL must be set and the server's security must be active (EXTSEC=ON).

For CA-ACF2, a new SAFPROT record is needed:

```
CNTLPT(SSCON) SUBSYS(SSCON) CLASSES(FACILITY VERIFY)
CNTLPT(SSCON) SUBSYS(SVC019) CLASSES(DATASET)
```

If you are using CA-ACF2 and do not have a class named FACILITY, you can select another class. You must customize all of the console specifications described previously.

Configuring IMS/DBCTL Access

If you configure for DBCTL access in EDACFGF/H, you must edit the JCL in *servertype*JCL. You must uncomment and supply a valid FOCPSB file. For information about creating the FOCPSB file, see *Customizing the IMS/DBCTL Environment* in Chapter 3, *Configuring Data Adapters*. See the IMS section of your *iWay Server Administration for MVS and VM* manual for more information on the FOCPSB file.

If you did not use the EDACFGx procedures, you must edit *servertype*SERV that you created in order to complete the IMS/DBCTL configuration. For details, see *How to Create Member Names in qualif.INSTALL.DATA* on page 8-3.

```
************************************************************************
**                        IMS DBCTL Keywords                          **
************************************************************************
IMSPZP  = xx
DBCTL   = START
IMSSEC  = {ON|OFF}
IMCLASS = PCICSPSB
************************************************************************
**                      End of IMS DBCTL Keywords                     **
************************************************************************
```

where:

xx

Is the suffix of the DFSPZP module created during the generation of the IMS/DBCTL Data Adapter.

ON

Activates DBCTL security. This setting requires that the server be in authorized mode.

OFF

Does not implement DBCTL security. This value is the default.

PCICSPSB

Points to a valid class that contains the security rule. The system administrator can define any class name required by the security implementation of the site. The default value is PCICSPSB.

Note: Do *not* change DBCTL=START.

Configuring Services for Inbound Protocols

When a client application connects to a server, it supplies the name of the service block to which it is connecting. A service block contains the parameters for a specific function of the server.

Syntax

How to Add a Service for LU6.2, and TCP/IP Inbound Protocols

Append the file *qualif*.EDACTL.DATA(CS3SRV) to *servertype*SERV.

Note: Pooled deployment parameters are only valid for users connecting with LU6.2 or TCP/IP.

```
*****************************************************************
**                 Generic Service Block                        *
*****************************************************************
SERVICE           = service
PROGRAM           = TSCOM3
***************   Pooled Deployment parameters   ***************
*DEPLOYMENT       = POOLED
*REFRESH_LIMIT    = 100
***********   End of Pooled Deployment parameters   ************
MAXIMUM           = nnn
TRUST_NT          = N
SERVINIT          = *,++
*********   Application Name Space parameter SERVINIT  *********
*APPROOT          = approotvalue
****   End of Application Name Space parameters SERVINIT   *****
***********    Pooled Deployment parameters SERVINIT   *********
*USER             = userid
*PASSWORD         = password
*******   End of Pooled Deployment parameters SERVINIT   *******
  DYNAM ALLOC FILE EDASPROF -
        DA qualif.EDAPROF.DATA(profile) SHR REU
  DYNAM ALLOC FILE EDAPROF  -
        DA qualif.EDAPROF.DATA          SHR REU
*****************************************************************
**        The following allocation is for tracing              **
*****************************************************************
* DYNAM ALLOC FILE IBITRACE -
*       DA qualif.EDACTL.DATA(IBITRACE) SHR REU
++
```

where:

service

Is the logical name for this service. It must be a unique name within the EDASERVE file in *servertype*SERV. The default value is EDASERVE.

deployment

The deployment type for this server. The modes of deployment are:

PRIVATE: (default) Each user of the server receives a private task, which will be cleared when they disconnect so the task will be reused by the next user.

Note: PDS indexes will not be reread in this mode. This is not true for server profiles, which are always refreshed for each user connection.

PRIVATE,REREAD: Same as for PRIVATE but all PDS files will be reread for each user. Should only be used for EDADBA service or when in development mode.

ISOLATED: Each user of the server will get a private task, which will be deleted when they disconnect from the server.

POOLED: Each user of the server reuses the task and context of the last user of the task. The server does not delete tasks when the user disconnects.

refresh_limit

Is the number of connections you allow an application agent before the server refreshes the agent's environment. After this specified number of client connections, the server terminates and then restarts the agent. The default value is 100. Uncomment this line for both private and pooled deployment.

nnn

Is the maximum number of users allowed for the service.

userid

Is the user ID for the application agents in a pooled environment. In a pooled environment, the tasks assume the identity of the user ID and password supplied during configuration.

password

Is the password for the user ID, specified for the application agents in a pooled environment.

qualif

Is the high-level qualifier for the data sets.

profile

Is the member name of the profile. It can be a new member or you can use the profile created during server configuration. For a Hub Server, the member name is H*owner_id;* and for a Full-Function Server, it is F*owner_id*. For a DBA service block, the member name is M*owner_id*.

Note: If the user ID and password parameters are left commented and pooled deployment is selected, the user ID of the person who submitted the server job is used as the security profile for all users of this service.

Example Adding a Service for LU6.2 or TCP/IP

The following is an example of a service for LU6.2 or TCP/IP:

```
*****************************************************************
*   Full Function Data Adapter
*   Global Configuration File
*   Generated on 2 Oct 2001 at 16:37:55
*   The lowercase parameters on the right of the equals sign
*   are site specific and need to be modified during the
*   configuration of the Data Adapter.
EXTSEC      = OFF
*SMFNUM     = 0
FASTLOAD    = **AUTO**
LONGSYNM    = ON
LICENSE     = 100-111-0422
****************   Optional Console Parameters   ****************
*
*CONSEC     = INTERNAL
*CONSOPER   = userid1,userid2
*CONSDTL    = userid1
*CONSUSER   = *
*
****************   End of Console Parameters   *****************
*****************************************************************
**                Generic Service block                        **
*****************************************************************
SERVICE        = EDAUSER
PROGRAM        = TSCOM3
**************    Pooled Deployment parameters   ****************
*DEPLOYMENT    = POOLED
*REFRESH_LIMIT = 100
***********   End of Pooled Deployment parameters   *************
MAXIMUM        = 16
TRUST_NT       = N
SERVINIT       = *,++
***********   Pooled Deployment parameters SERVINIT      ********
*USER          = userid
*PASSWORD      = password
*******   End of Pooled Deployment parameters SERVINIT   ********
  DYNAM ALLOC FILE EDASPROF -
        DA EDAARH.V5R1M00.EDAPROF.DATA(FEDAARH) SHR REU
  DYNAM ALLOC FILE EDAPROF  -
        DA EDAARH.V5R1M00.EDAPROF.DATA          SHR REU
*********************   Server Tracing   ***********************
*
* DYNAM ALLOC FILE IBITRACE -
*       DA EDAARH.V5R1M00.INSTALL.DATA(IBITRACE)  SHR REU
****************   End of Server Tracing   **********************
```

```
++
*****************************************************************
**                 EDADBA  Service block                       **
*****************************************************************
SERVICE        = EDADBA
PROGRAM        = TSCOM3
MAXIMUM        = 1
SERVINIT       = *,++
 DYNAM ALLOC FILE EDASPROF -
        DA EDAARH.V5R1M00.EDAPROF.DATA(MEDAARH) SHR REU
**********************   Server Tracing   **********************
*
* DYNAM ALLOC FILE IBITRACE -
*       DA EDAARH.V5R1M00.INSTALL.DATA(IBITRACE)  SHR REU
****************   End of Server Tracing   **********************
++
*****************************************************************
**                  Generic Service Block                      **
*****************************************************************
SERVICE        = EDASERVE
PROGRAM        = TSCOM3
***************   Pooled Deployment parameters   ****************
*DEPLOYMENT    = POOLED
*REFRESH_LIMIT = 100
***********   End of Pooled Deployment parameters   *************
MAXIMUM        = 16
TRUST_NT       = N
SERVINIT       = *,++
***********   Pooled Deployment parameters SERVINIT      ********
*USER          = userid
*PASSWORD      = password
*******   End of Pooled Deployment Parameters SERVINIT   ********
  DYNAM ALLOC FILE EDASPROF -
        DA EDAARH.V5R1M00.EDAPROF.DATA(NEWPROF) SHR REU
  DYNAM ALLOC FILE EDAPROF  -
        DA EDAARH.V5R1M00.EDAPROF.DATA          SHR REU
*****************************************************************
**        The following allocation is for tracing              **
*****************************************************************
* DYNAM ALLOC FILE IBITRACE -
*       DA EDAARH.V5R1M00.EDACTL.DATA(IBITRACE) SHR REU
++
```

Modifying the Communications Subsystem Configuration File

The major components of the communications subsystem configuration file are:

- Global keywords. See *Keywords in a Communications Subsystem Configuration File* on page 8-21.
- Protocol block(s). See *Protocol and Node Blocks in a Communications Subsystem Configuration File* on page 8-22.
- Node block(s). See *How to Add a Node Block for All Protocols* on page 8-24.

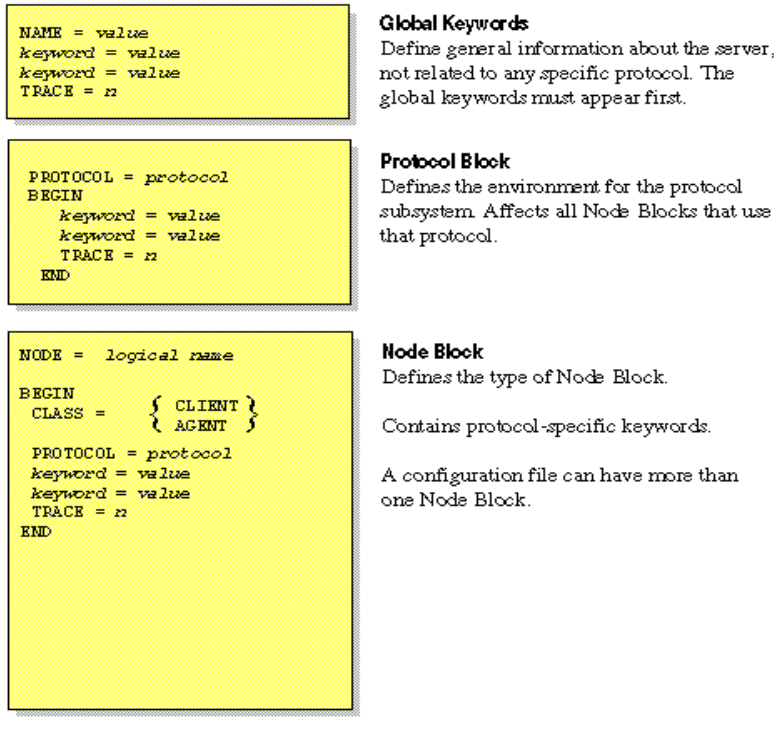
  Node blocks control either *inbound* or *outbound* communications. CLASS=AGENT node blocks control inbound communications, and CLASS=CLIENT node blocks control outbound communications.

For an illustration that defines these components, see *Communications Subsystem Configuration File Structure* on page 8-20.

Reference

Communications Subsystem Configuration File Structure

The following figure shows the basic construction of the communications subsystem configuration file.



Reference BEGIN-END Phrases in a Communications Subsystem Configuration File

Keyword-value pairs within a protocol block or node block must be enclosed between BEGIN-END phrases, as follows:

```
BEGIN
    keyword = value
    keyword = value
END
```

Global keywords such as NAME or TRACE are exceptions; these keywords belong at the beginning of the configuration file, outside the scope of BEGIN-END phrases.

Reference Comments in a Communications Subsystem Configuration File

You can enter a comment on any line in the configuration file by using a semicolon (;). Any text appearing to the right of and including a semicolon is a comment. Comments can share the same line as keywords. For example:

```
NAME = xyz machine on 4th floor ; This is a comment.
```

Reference Keywords in a Communications Subsystem Configuration File

The following section describes the global keywords of the communications subsystem configuration file

```
NAME        = name value
QUEUE DEPTH = queue depth value
DEMAND LOAD = demand load
TRACE       = trace value
```

where:

name value

Can be up to 32 characters, must begin with a letter (A-Z), and can include any characters except a semicolon (;) and equal sign (=). You include the keyword, NAME, at the beginning of the configuration file, make it unique throughout the entire network, and make the name as descriptive as possible. The default value is UNKNOWN.

The NAME keyword identifies the device throughout the entire network, encompassing all platforms and protocols.

queue depth value

Is the amount of memory used (QUEUE DEPTH multiplied by 16K). The default value is 8, which means that the communications system never uses more than 0.5 megabytes of memory. A QUEUE DEPTH of 8 is recommended. Changes to this value from the value of 8 supplied in the sample configuration files should be based on a clear understanding of the consequences, in the context of the workload the communications system will service. A QUEUE DEPTH of 8 is a good compromise between memory usage and speed.

The QUEUE DEPTH keyword is a tuning parameter that determines the maximum amount of memory the communications system can use to store records before they are returned to the client.

demand load

Determines how the communications files are loaded. Possible values are:

ON loads the files when they are needed. This is the default.

OFF loads all of the files upon initialization. Use this option if you plan to issue rapid connects and disconnects.

trace value

The possible trace levels are:

0 for no tracing.

1 for error messages (the default).

2 for warnings.

4 for information.

8 for detailed information.

16 for data buffer.

The values are cumulative, that is, you can set TRACE = 3 (1+2) to see error messages and warnings. To see everything, set TRACE = 31. The default value for error messages is 1.

Note: This is the only place where the communications trace can be activated.

Protocol and Node Blocks in a Communications Subsystem Configuration File

The protocol block defines the environment for the protocol subsystem. This definition affects all node blocks that use the specified protocol.

For the TCP/IP and EMC protocols, the protocol block provides information about the protocol subsystem being used. Only one protocol block per communications configuration file is allowed. For details, see *How to Add a Protocol Block for TCP/IP* on page 8-23 and *How to Add a Node Block for All Protocols* on page 8-24.

Syntax How to Add a Protocol Block for TCP/IP

```
PROTOCOL      = TCP
  BEGIN
    TCP NAME = tcp name
    VENDOR   = vendor name
  END
```

where:

tcp name

For IBM TCP/IP is the name of the started task or the JOB name for the TCP/IP subsystem address space, if it is different from the default (TCPIP).

For Interlink TCP/IP is the name of the TCP/IP subsystem (no default value).

For OCS use the default value of TCPIP.

Note: The server can only support one TCP/IP vendor at one time for both inbound and outbound blocks. Use multiple servers to receive support from multiple vendors.

vendor name

Is the TCP/IP vendor name. Possible values are IBM, INTERLINK, or OCS.

Example Adding a Protocol Block for TCP/IP

The following is an example of a protocol block for IBM TCP/IP:

```
PROTOCOL          = TCP
  BEGIN
    TCP NAME      = TCPIP
    VENDOR        = IBM
    REUSEADDR    = ON
  END
```

The following is an example of a protocol block for INTERLINK TCP/IP:

```
PROTOCOL      =   TCP
  BEGIN
    TCP NAME = ACSS
    VENDOR   = INTERLINK
  END
```

Syntax How to Add a Node Block for All Protocols

The node block is used for all protocols. The syntax for the node block is as follows

```
NODE         = logical name
  BEGIN
    PROTOCOL = protocol value
    CLASS    = class value [(service)]
    keyword  = value
    other keywords
  END
```

where:

logical name

Identifies the logical name of the node block on which the server is receiving or sending data.

Logical name can be up to 8 characters, must begin with a letter (A-Z), and can include any characters except a semicolon (;) or an equal sign (=). This name is user-defined and must be unique in this configuration file.

protocol value

Defines the protocol subsystem being used. This keyword is required.

Each protocol requires different protocol-specific keywords, which are defined in the following sections. The possible values are LU62 and TCP.

class value

Defines this node block as a Synchronous client node block or an Agent node block. The possible values are as follows:

`CLIENT` defines this node block as a Synchronous Client node block, which defines how the device sends outbound communications to a subserver. The value for *service* determines the service name that is executed on the subserver (MVS only). The default value of *service* is the value specified for *logical name* for this node block.

`AGENT` defines this node block as an Agent node block, which defines how the server receives inbound communications from another source (client, gateway, or server).

other keywords

Depend on the value for PROTOCOL and CLASS.

Configuring Inbound Communications

The server supports multiple inbound protocol connections. To add an inbound protocol to your configuration, copy an inbound (server) communications file to *servertype*CSG and edit it for your site.

You can build on the same *servertype*CSG to support multiple inbound protocol connections.

Syntax

How to Add an LU6.2 Inbound Block

Append the file *qualif*.EDACTL.DATA(CS3LU6I) to *servertype*CSG. The CS3LU6I file is shown below

```
NODE                 = LU62IN
  BEGIN
    PROTOCOL         = LU62
    CLASS            = AGENT
    LOCAL LU NAME    = local_lu_name    ;Applid server will listen on
    TP NAME          = MVSSRVR          ;Arbitrary name (e.g., MVSSRVR)
    CONNECT_INTERVAL = nn
  END
```

where:

local_lu_name

Is the APPLID that the server will listen on for inbound requests.

nn

Is the time, in seconds, that a connect request will be queued before it is rejected. A value of 0 (zero) will disable queuing.

If a connect request fails (or a queued connect request exceeds the set time period), an EDAAPI -33 error code will be returned to the application program.

You must enter all values in uppercase.

For an example, see *Adding an LU6.2 Inbound Block* on page 8-26. For more information on connection queuing, see the *iWay Server Administration for MVS and VM* manual.

Example Adding an LU6.2 Inbound Block

The following is an example of an LU6.2 inbound block:

```
NAME                  = Server for MVS release 5.X
QUEUE DEPTH           = 8
;TRACE                = 31
NODE                  = LU62IN
  BEGIN
    PROTOCOL          = LU62
    CLASS             = AGENT
    LOCAL LU NAME     = ED38APP1    ;Applid server will listen on
    TP NAME           = MVSSRVR     ;Arbitrary name (for example, MVSSRVR)
    CONNECT_INTERVAL  = 10
  END
```

Note: Connection queuing has been added to the LU6.2 inbound requests.

Syntax How to Add a TCP/IP Inbound Block

Append the file *qualif*.EDACTL.DATA(CS3TCPI) to *servertype*CSG. The CS3TCPI file is shown below

```
; This Protocol block is needed to find the started task for TCP/IP.
; For IBM's TCP, the default name is TCPIP.
; For Interlink's TCP, this is the name of the subsystem (no default).
;
PROTOCOL              = TCP
  BEGIN
    TCP NAME          = tcp name
    VENDOR            = vendor name
    ;REUSEADDR        = ON
    ;TCPIP_METHOD     = IUCV
  END
NODE                  = TCPIN
  BEGIN
    PROTOCOL          = TCP
    CLASS             = AGENT
    SERVICE           = nnnn          ;Port number the server will use
    KEEPALIVE         = keepalive
    ATM_INTERVAL      = atm_interval value
    ATM_MAXTRIES      = atm_maxtries value
    CONNECT_INTERVAL  = nn
  END
```

where:

tcp name

For IBM TCP/IP is the name of the started task or the JOB name for the TCP/IP subsystem address space, if it is different from the default (TCPIP).

If you are accessing your server through IBM TCP/IP, and your TCP/IP JOBNAME is not TCPIP, you may be unable to connect to the server. You will see several messages in your server job in the form:

```
LSCXNNN **** WARNING **** ERRNO = ERRORTYPE
```

To correct this, copy the DD card //SYSTCPD from your TCP/IP JCL and add it to your server JCL, pointing to the same data set as it is in the TCP/IP job.

If you are using IBM TCP/IP Version 3.2, the //SYSTCPD allocation must be added to the server JCL.

For Interlink, TCP/IP is the name of the TCP/IP subsystem (no default value). You must include VENDOR=INTERLINK in the protocol block.

For OCS, use the default value of TCPIP.

Note: The server can only support one TCP/IP vendor at one time for both inbound and outbound blocks. Use multiple servers to receive support from multiple vendors.

vendor name

Is the TCP/IP vendor name. Possible values are IBM, INTERLINK, or OCS.

`;REUSEADDR=ON`

For IBM TCP/IP Communications Server (OS/390 2.5 or higher) ONLY.

Uncomment this parameter if you are using an IBM TCP/IP Communications Server and a restart of the server causes the TCP attach manager to enter a re-try sequence. This is caused by TCP/IP not releasing the port at the time the server is stopped. The use of this parameter will free the port when the server is stopped so it can be restarted immediately.

;TCPIP_METHOD=IUCV

For IBM TCP/IP Version 3.1 or 3.2 (OS/390 2.4 or earlier) ONLY.

Uncomment this parameter if you are using IBM TCP/IP Version 3.1 or 3.2 and the user ID who submitted the server task has an Open Edition directory (OE segment).

nnnn

Is the port name or number used by the server to listen for client requests. It is suggested that the system administrator assign the number 3000 or greater.

Note: The server supports port numbers with four or five digits, as well as Domain Name Services.

keepalive

(Agent and Gateway Node Blocks only) Causes Attach Manager to check for the availability of the client at intervals that coincide with the corresponding KEEPALIVE interval parameter for your system's TCP/IP communications. If the client is not available, then Attach Manager brings down the unattached agent process. Turning on KEEPALIVE enables the server to free a resource that may be left in use when a client abruptly disconnects from the server. Possible values are:

`0` turns off KEEPALIVE (the default).

`1` turns on KEEPALIVE.

n is for Interlink TCP only: n>15 where n is the number of minutes.

atm_interval value

Is the number of minutes the server will wait before attempting to acquire the resource. The default value is 5 minutes; the minimum value is 5.

atm_maxtries value

Is the number of attempts the server will make to acquire the resource. The default value is 12 attempts; the maximum value is 12.

nn

Is the time, in seconds, that a connect request will be queued before it is rejected. A value of 0 (zero) will disable queuing.

If a connect request fails (or a queued connect request exceeds the set time period), an EDAAPI -33 error code will be returned to the application program.

You must enter all values in uppercase.

For an example, see *Adding a TCP/IP Inbound Block* on page 8-29.

Example Adding a TCP/IP Inbound Block

Assume that the server uses IBM TCP/IP, so that the value for the started task name on the Sample Configuration Worksheet is the same as the default.

```
NAME                  = Server for MVS release 5.X
QUEUE DEPTH           = 8
;TRACE                = 31
NODE                  = LU62IN
  BEGIN
    PROTOCOL          = LU62
    CLASS             = AGENT
    LOCAL LU NAME     = ED38APP1    ;Applid server will listen on
    TP NAME           = MVSSRVR     ;Arbitrary name (for example, MVSSRVR)
    CONNECT_INTERVAL  = 10
  END
; This Protocol block is needed to find the started task for TCP/IP.
; For IBM's tcp, the default name is TCPIP.
; For Interlink's tcp, this is the name of the subsystem (no default).
;
PROTOCOL              = TCP
  BEGIN
    TCP NAME          = TCPIP
    VENDOR            = IBM
    REUSEADDR         = ON
  END
NODE                  = TCPIN
  BEGIN
    PROTOCOL          = TCP
    CLASS             = AGENT
    SERVICE           = MYPORT1          ;Port number the server will use
    ATM_MAXTRIES      = 3
  END
```

Note: ATM MAXTRIES had been added to the protocol block. The retry time is now 15 minutes (5 X 3).

If you are using Interlink, you must add VENDOR=INTERLINK in the protocol block. The following is a sample EDACSG:

```
NAME                  = Server for MVS release 5.X
QUEUE DEPTH           = 8
;TRACE                = 3
NODE                  = LU62IN
  BEGIN
    PROTOCOL          = LU62
    CLASS             = AGENT
    LOCAL LU NAME     = ED38APP1    ;Applid server will listen on
    TP NAME           = MVSSRVR     ;Arbitrary name (for example, MVSSRVR)
    CONNECT_INTERVAL  = 10
  END
; This Protocol block is needed to find the started task for TCP/IP.
; For IBM's tcp, the default name is TCPIP.
; For Interlink's tcp, this is the name of the subsystem (no default).
;
PROTOCOL              = TCP
  BEGIN
    TCP NAME          = ACSS
    VENDOR            = INTERLINK
  END
NODE                  = TCPIN
  BEGIN
    PROTOCOL          = TCP
    CLASS             = AGENT
    SERVICE           = MYPORT2          ;Port number the server will use
    ATM_MAXTRIES      = 3
  END
```

Configuring Outbound Communications

The process of configuring outbound communications pertains *only* to Hub Servers or Full-Function Servers with Hub Services.

The server supports multiple outbound protocol connections to any number of Subservers. To add an outbound protocol to your configuration, you must copy the applicable outbound (server) communications file to *servertype*CSG.

You can build on the same *servertype*CSG to support multiple outbound protocol connections. You must enter all values in uppercase.

Syntax

How to Match the Outbound Block to the Subserver

The general syntax of the outbound block, and how the outbound block on the Hub matches the inbound block on the subserver, is as follows:

```
NODE                = logical name
  BEGIN
    PROTOCOL        = protocol
    CLASS           = CLIENT[(logical name override)]
    protocol specific keywords
    ...
  END
```

Note: You must enter all values in uppercase.

Reference

Requirements for Connecting to a SubServer at the Protocol Level

When connecting to any Subserver, a connection at the protocol level (physical level) is established first. For this to be successful, the following must be true:

- Both servers *must* be using the same protocol.
- The following protocol resources *must* match:
  - For TCP, the IP addresses and port numbers.
  - For LU6.2, the PARTNER LU NAME on the Hub Server and the LOCAL LU NAME on the subserver, in addition to the TP NAME on both servers.

Once the physical level connection has been made, a logical level connection is established. If you are connecting to a subserver other than MVS, this logical level connection is automatic.

For MVS, the logical level connection is successful only if either of the following is true:

- The NODE logical name matches a SERVICE name of the EDASERVE configuration file of the subserver.

  OR (optional)

- The CLASS=CLIENT override matches a SERVICE name of the EDASERVE configuration file of the subserver. The CLASS=CLIENT logical name override supersedes the NODE logical name in the matching process.

If you are connecting to a UNIX, NT, OpenVMS, or OS/400 subserver, the value for logical name can be any 1-8 character descriptive name as no matching takes place.

Syntax How to Add an LU6.2 Outbound Block

Append the file *qualif*.EDACTL.DATA(CS3LU6O) to *servertype*CSG. CS3LU6O is shown below

```
NODE                = logical name
  BEGIN
    PROTOCOL        = LU62
    CLASS           = CLIENT
    LOCAL LU NAME   = local_lu_name     ;Unique applid for each client
    PARTNER LU NAME  = partner_lu_name   ;Must match the server's local
lu name
    TP NAME         = tp_name           ;Must match the server's TP NAME
    MODE NAME       = entry             ;Log mode entry for local applid
  END
```

where:

logical name

Change the logical name value. For details, see *How to Match the Outbound Block to the Subserver* on page 8-31.

You can also use the CLASS=CLIENT override. For a definition, see *How to Add a Node Block for All Protocols* on page 8-24.

partner_lu_name

Is a unique APPLID used to identify this outbound connection to VTAM. This method allows only one client request to be passed to the subserver pointed to by this outbound block. If more than one client will be requesting data from the subserver, you should set this value to (LU6POOL). If you use this option, allocate the ddname LU6POOL, in the server JCL, to point to a list of APPLIDs. When the Hub Server connects to the subserver, it chooses the first unused APPLID from the list to uniquely identify itself to VTAM.

tp_name

Is an internal name used by communications for LU6.2. This value must match the value for TP NAME specified in the subserver's inbound LU6.2 block.

entry

Is the log mode table entry name the local APPLID was defined with.

You must enter all values in uppercase.

For an example, see *Adding an LU6.2 Outbound Block* on page 8-33.

Example Adding an LU6.2 Outbound Block

```
NAME                  = Server for MVS release 5.X
QUEUE DEPTH           = 8
;TRACE                = 31
PROTOCOL              = TCP
  BEGIN
    TCP NAME          = TCPIP
    VENDOR            = IBM
    REUSEADDR         = ON
  END
NODE                  = TCPIN
  BEGIN
    PROTOCOL          = TCP
    CLASS             = AGENT
    SERVICE           = MYPORT1
  END
NODE                  = MVSSUB2
  BEGIN
    PROTOCOL          = LU62
    CLASS             = CLIENT
    LOCAL LU NAME     = ED38APP2          ;Unique applid for each client
    PARTNER LU NAME   = ED38APP1           ;Must match the server's local
lu name
    TP NAME           = MVSSRVR           ;Must match the server's TP NAME
    MODE NAME         = PARALLEL          ;Log mode entry for local applid
  END
```

Syntax How to Add TCP/IP Outbound

Append the file *qualif*.EDACTL.DATA(CS3TCPO) to *servertype*CSG. CS3TCPO is shown below:

```
; This Protocol block is needed to find the started task for TCP/IP.
; For IBM's tcp, the default name is TCPIP.
; For Interlink's tcp, this is the name of the subsystem (no default).
;
;PROTOCOL          = TCP
;  BEGIN
;    TCP NAME      = TCPIP
;    VENDOR        = IBM
;  END
NODE                = logical name
  BEGIN
    PROTOCOL        = TCP
    CLASS           = CLIENT
    HOST            = ip_address        ;IP address of host server
    SERVICE         = nnnn              ;Port # server is listening on
  END
```

where:

`logical name`

Change the logical name value. For details, see *How to Match the Outbound Block to the Subserver* on page 8-31.

You can also use the CLASS=CLIENT override. For a definition, see *How to Add a Protocol Block for TCP/IP* on page 8-23.

`ip_address`

Is the IP address of the subserver. Enter either the numeric IP address or the host name representing the IP address.

`nnnn`

Is the port the subserver is listening on. Enter either the port number or the service name representing the port number.

Note:

By default, the protocol block of the TCP outbound sample (CS3TCP0) is commented out. If this is the first occurrence of a protocol block in the communications configuration file, uncomment the protocol block and supply appropriate values for TCP NAME and VENDOR. For more information on TCP NAME and VENDOR, see *How to Add a Protocol Block for TCP/IP* on page 8-23.

You must enter all values in uppercase.

For illustrations, see *Using IBM TCP/IP Outbound* on page 8-35 and *Using Interlink TCP/IP Outbound* on page 8-36.

Example Using IBM TCP/IP Outbound

```
NAME                  = Server for MVS release 5.X
QUEUE DEPTH           = 8
;TRACE                = 31
PROTOCOL              = TCP
  BEGIN
    TCP NAME          = TCPIP
    VENDOR            = IBM
    REUSEADDR         = ON
  END
NODE                  = TCPIN
  BEGIN
    PROTOCOL          = TCP
    CLASS             = AGENT
    SERVICE           = MYPORT1
  END
NODE                  = MVSSUB2
  BEGIN
    PROTOCOL          = LU62
    CLASS             = CLIENT
    LOCAL LU NAME     = ED38APP2          ;Unique applid for each client
    PARTNER LU NAME = ED38APP1            ;Must match the server's local lu
name
    TP NAME           = MVSSRVR           ;Must match the server's TP NAME
    MODE NAME         = PARALLEL          ;Log mode entry for local applid
  END
NAME    = iWay MVS 3.x Client Using CS/3
; This Protocol block is needed to find the started task for TCP/IP.
; For IBM's tcp, the default name is TCPIP.
; For Interlink's tcp, this is the name of the subsystem (no default).
;
;PROTOCOL             = TCP
;  BEGIN
;    TCP NAME         = tcp name
;  END
NODE                  = UNIXSUB1
  BEGIN
    PROTOCOL          = TCP
    CLASS             = CLIENT
    HOST              = UNIX20            ;IP address of host server
    SERVICE           = 3333              ;Port # server is listening on
  END
```

Example Using Interlink TCP/IP Outbound

```
NAME                  = Server for MVS release 5.X
QUEUE DEPTH           = 8
;TRACE                = 31
NODE                  = LU62IN
  BEGIN
    PROTOCOL          = LU62
    CLASS             = AGENT
    LOCAL LU NAME     = ED38APP1     ;Applid server will listen on
    TP NAME           = MVSSRVR      ;Arbitrary name (for example, MVSSRVR)
    CONNECT_INTERVAL  = 10
  END
NODE                  = MVSSUB2
  BEGIN
    PROTOCOL          = LU62
    CLASS             = CLIENT
    LOCAL LU NAME     = ED38APP2     ;Unique applid for each client
    PARTNER LU NAME   = ED38APP3     ;Must match the server's local lu name
    TP NAME           = MVSSRVR      ;Must match the server's TP NAME
    MODE NAME         = PARALLEL     ;Log mode entry for local applid
  END
; This Protocol block is needed to find the started task for TCP/IP.
; For IBM's tcp, the default name is TCPIP.
; For Interlink's tcp, this is the name of the subsystem (no default).
;
PROTOCOL              = TCP
  BEGIN
    VENDOR            = INTERLINK
    TCP NAME          = ACSS
    END
NODE                  = MVSSUB2
  BEGIN
    PROTOCOL          = TCP
    CLASS             = CLIENT
    HOST              = MVSLINK1     ;IP address of host server
    SERVICE           = 4321         ;Port # server is listening on
  END
```

Note: Because this example shows Interlink TCP, the protocol block of the outbound TCP sample code (CS3TCP0) has been uncommented. It is the first occurrence of a protocol block in the communications configuration file.

CHAPTER 9

Accessing the Server

Topics:

- Starting the Server
- Stopping the Server

The server must be started before you can perform any tasks. For instructions, see *Starting the Server* on page 9-2 and *Stopping the Server* on page 9-2.

Starting the Server

Once you start the IBI Subsystem, you can start the server by submitting the server JCL as a started task or a batch job.

Upon successful initialization of the server, the following message is written to both the server EDAPRINT message log and the MVS console log:

```
(EDA13023) ALL INITIAL SERVERS STARTED
```

Once the server starts, it should be allowed to remain active indefinitely, or as required by your administrative staff. The only CPU resource consumed by a server job not processing a request is periodic calls to the operating system by the TIMER function. The TIMER function monitors resource use and terminates idle tasks. This overhead is minuscule, and has no impact on your MVS system's performance.

You can now start the Console and issue operator commands. For more information, see Chapter 11, *Using the Server Console*.

Stopping the Server

The server provides MVS systems operators with the ability to stop the server job. However, when you terminate the server job, all requests (both pending and executing) are terminated, and all request output not yet received by connected clients is lost. The MVS operator command, F (modify), terminates the Server for MVS.

Syntax **How to Shut Down the Server**

The syntax of the MVS modify command to terminate the server is

```
/F jobname,STOP
```

where:

jobname

Is the server job name or started task name.

Example **Stopping the Server**

The following command terminates processing by all active users and shuts down the EDASERV server:

```
/F EDASERV,STOP
```

CHAPTER 10

IBI Subsystem

Topics:

- Operating the IBI Subsystem With the SUBSYSI Program
- Interacting With the IBI Subsystem
- Controlling the Hiperspace Usage of the IBI Subsystem
- Troubleshooting

The IBI Subsystem is a state-of-the-art facility that provides communications between and coordination among address spaces running server products on MVS systems. It expands storage control, enhances error recovery capability, provides new control facilities, and improves performance and capacity.

For related information, see *Installing the IBI Subsystem* in Chapter 5, *Configuring Server System Features*.

Operating the IBI Subsystem With the SUBSYSI Program

The SUBSYSI program provides control of the IBI Subsystem. Its basic function is to initialize the subsystem. It can also be used to stop, unload, and replace the subsystem modules in storage.

SUBSYSI may be run as a batch job or as a started task. JCL samples are provided in members SUBSYSIJ and SUBSYSP, respectively, of *qualif*.EDALIB.DATA.

You should not use SUBSYSI to STOP or REMOVE the subsystem while any server address spaces are running. All servers must be stopped before stopping the subsystem. While HiperEDA users may continue to run, the HiperBUDGET feature will be deactivated when the subsystem is stopped.

Note: You must use the subsystem supplied with this release. You cannot use this release of the server with older subsystem code.

Syntax

How to Use the SUBSYSI Program

Valid parameters of SUBSYSI are:

- START: Initialize the subsystem—load modules into CSA. If the modules have already been loaded but the subsystem is not active (for example, STOPped), RESTART is assumed and executed.

  You can provide a second parameter after the START if you are using a CPF macro as a prefix. The syntax is:

  ```
  START, prefix
  ```

  There should be no spaces around the comma. If the prefix is used, the status of the subsystem can be displayed anywhere in a sysplex. If the prefix value is MVSA, you can issue the operation command MVSA DISPLAY SUBSYSTEM and the result will be displayed on the MVS console through which the command was issued. For more information on the CPF macro, see your IBM documentation.
- STOP: Logically disable the subsystem without removing the modules from CSA. Do not do this while any servers are running.
- RESTART: Restart the subsystem following the STOP command. The modules already exist in CSA (for example, START has already been run).
- REPLACE: Reload modules into CSA. This is used to upgrade without re-IPLing MVS. You should only run this after performing a STOP.
- REMOVE: Disable the subsystem and remove modules from CSA. This operation leaves the SSCT entry and small (2K) module, but the module will be inactive. Do not do this while any servers are running.

When START or RESTART is specified, SUBSYSI reads commands from SYSIN, as if they were issued from the MVS console. This is useful for setting the HiperBUDGET limits from a batch job or started task. A sample is provided in *qualif*.EDACTL.DATA(SUBSYSIH).

Interacting With the IBI Subsystem

You can control the IBI Subsystem from the TSO command line, where you can issue operator and display commands.

Syntax

How to Issue Operator Controls to the IBI Subsystem

To issue operator commands to the subsystem from the TSO command line, prefix them with a slash (/) and the IBI Subsystem name. There may be no leading blanks in the command, but extra blanks are permitted in the command text. For example, assuming the subsystem has been installed as "IBIS," the following commands are all valid:

```
/IBIS D SUBSYSTEM
/IBIS   DISPLAY   IBIPROD
/IBIS            D IBIPROD
/IBIS SET  MVSLIM=500
/IBIS T FILELIM=300, MVSLIM=500
```

Submitting commands using the SUBSYSI program can also control the subsystem. In this case, the PARM parameter on the EXEC card should not be specified. The commands are supplied using the SYSIN DD.

Example

Submitting Commands Using the SUBSYSI Program

The following example shows how to submit commands using the SUBSYSI program. The output of these commands will be in the job log.

```
//SUBSYSI     EXEC PGM=SUBSYSI,REGION=4096K,
//STEPLIB     DD   DSN=qualif.EDALIB.LOAD,DISP=SHR
//SYSIN       DD   *
D SUBSYSTEM
D IBIPROD
D HIPERBUDGET
/*
```

Displaying IBI Subsystem Information

The DISPLAY commands from the TSO command line show subsystem related or subsystem-managed information. The keyword DISPLAY may be abbreviated as "D". The following commands are available:

- DISPLAY IBIPROD displays one line of information for every server address space running on the system. For servers, information such as current and maximum number of users and LU names are displayed.
- DISPLAY HIPERBUDGET shows current expanded storage usage and HiperBUDGET limit values.
- DISPLAY SUBSYSTEM shows internal information about the subsystem.

Controlling the Hiperspace Usage of the IBI Subsystem

If you are using the HiperEDA option, which uses hiperspaces to store information, then you can use HiperBUDGET. HiperBUDGET uses the IBI subsystem to regulate and report the overall use of hiperspace on that system. It accomplishes this through the enforcement of predefined limits on hiperspace consumption at the system, server, user, and file levels, whereby the limits set at the lower levels cannot exceed those set at a higher level.

SET Commands

The SET commands, issued from the TSO command line, are used with the HiperBUDGET feature of HiperEDA to change the limit values. The keyword SET may be abbreviated as "T". More than one limit may be set on the same command:

```
/IBIS T FILELIM=300, MVSLIM=500
```

Values that may be set are:

```
SET MVSLIM = mvslim | -1
SET SERVLIM = servlim | -1
SET TCBLIM = tcblim | -1
SET FILELIM = filelim | -1
```

The value of -1 specifies that there is no limit checking for that particular parameter.

Syntax How to Set Limits for the IBI Subsystem

The following are the limits that you can set for the IBI Subsystem:

```
[/IBIS] SET MVSLIM = {mvslim|-1}
```

where:

mvslim

Is the maximum number of (4K) pages of hiperspace for all of the products on that MVS system.

-1

Specifies that there is no limit checking.

```
[/IBIS] SET SERVLIM = {servlim|-1}
```

where:

servlim

Is the maximum number of (4K) pages of hiperspace allowed for multiple users on a per-server basis.

-1

Specifies that there is no limit checking.

```
[/IBIS] SET TCBLIM = {tcblim|-1}
```

where:

tcblim

Is the maximum number of (4K) pages available for each individual user.

-1

Specifies that there is no limit checking.

This parameter is equivalent to SET HIPERSPACE=*hiperspace*. If both parameters are specified, the smaller value takes precedence.

```
[/IBIS] SET FILELIM = {filelim|-1}
```

where:

filelim

Is the maximum number of (4K) pages of hiperspace on an individual file basis.

-1

Specifies that there is no limit checking.

This parameter is equivalent to SET HIPERFILE=*hiperfile* under the IBI Subsystem. If both parameters are specified, the smaller value takes precedence.

Syntax How to View Limit Specifications and Hiperspace Usage Statistics

To report on the limits specified and the actual utilization of hiperspace, issue the following query from any valid connector environment that supports RPC calling:

```
EDAEXEC '? HBUDGET'
```

Among the statistics displayed will be the limits set at the system, server, user and file levels, the number of busy pages, the number of hiperextents allowed, and the ddname and size of files allocated in hiperspace or “spilled” to disk.

Example Displaying Limit Specifications and Hiperspace Usage Statistics

The following example displays the limits specified and the hiperspace usage statistics of an MVS system:

```
Total system     limit is not set
Total server     limit is not set
Total hiperspace limit is not set
Single file size limit is     524288 pages
Total amount of busy pages is    616 pages
Number of extents is set to      127
_____________________________________________________________________
DDname   :Reserved :Hiperspace : Spilled  :Spill DDn
```

Troubleshooting

In the event of a problem with the IBI Subsystem, you may be able to resolve or isolate the problem before contacting our technical support staff. The most common causes of problems with the IBI Subsystem are:

- One or both of the zap jobs were not run, did not run correctly, or an incorrect value was used for the zap.
- The zap jobs were run against the wrong library.
- MVS was not IPLed, or the wrong system was IPLed, so the changes to SYS1.PARMLIB have not taken effect.
- The wrong SYS1.PARMLIB was updated, the wrong PARMLIB members were updated (they must be specified in the appropriate IEASYSxx), or the wrong system's PARMLIB was updated.
- SUBSYSI is not being run from an APF-authorized library.
- SUBSYSI was not run to START the subsystem.
- SUBSYSI was run with STOP or REMOVE and disabled the subsystem.
- For FOCUS Database Server, the communications dataset was not allocated.
- The communications dataset was allocated, but with the wrong ddname.
- The communications dataset was allocated incorrectly.
- There is a conflict with an existing subsystem name.
- There is a mismatch between the subsystem code and server code. If necessary, for a new level, you must re-run the appropriate copy and zap jobs.
- The server is running out of an unzapped library.
- The HiperBUDGET parameters were not properly set.

If, after considering these troubleshooting hints, you still need to call for assistance, see *Information You Should Have for Technical Support* on page 10-7.

Reference Information You Should Have for Technical Support

When contacting technical support for subsystem problems, the following information is helpful:

- For installation problems:
  - The JCL, input, and output for all zap and copy job runs.
  - All SYS1.PARMLIB updates performed.
- The output from the D SUBSYSTEM, D IBIPROD, and D HIPERBUDGET commands.
- The JCL, input, and output for all SUBSYSI job runs.
- The JCL and output for the failing address space(s).

CHAPTER 11

Using the Server Console

Topics:

- Accessing the Server Console
- Selecting Server Console Commands
- Executing MVS Operator Commands

Once you start the server, you can monitor and manage MVS server operations from the server console. See *Selecting Server Console Commands* on page 11-4.

You have the option of issuing several MVS operator commands, either from the server console or directly from the MVS operating system. For details, see *Executing MVS Operator Commands* on page 11-23.

Accessing the Server Console

Once the server is up and operational, you can enter the server console by logging onto the Console APPLID from any VTAM 3270 session.

Procedure How to Start the Server Console

1. Type the following command at the prompt

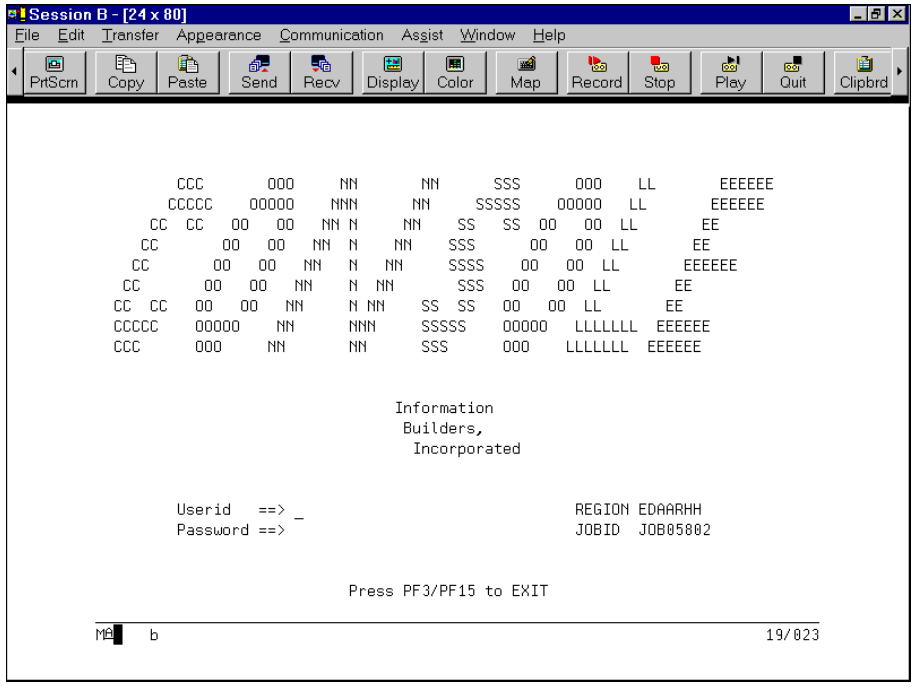
```
LOGON APPLID(consoleapplid)
```

where:

consoleapplid

Is the LU2 applid used to communicate with the console. You can find its value in the adapter JCL file (EDACONS).

The Console logon window opens:



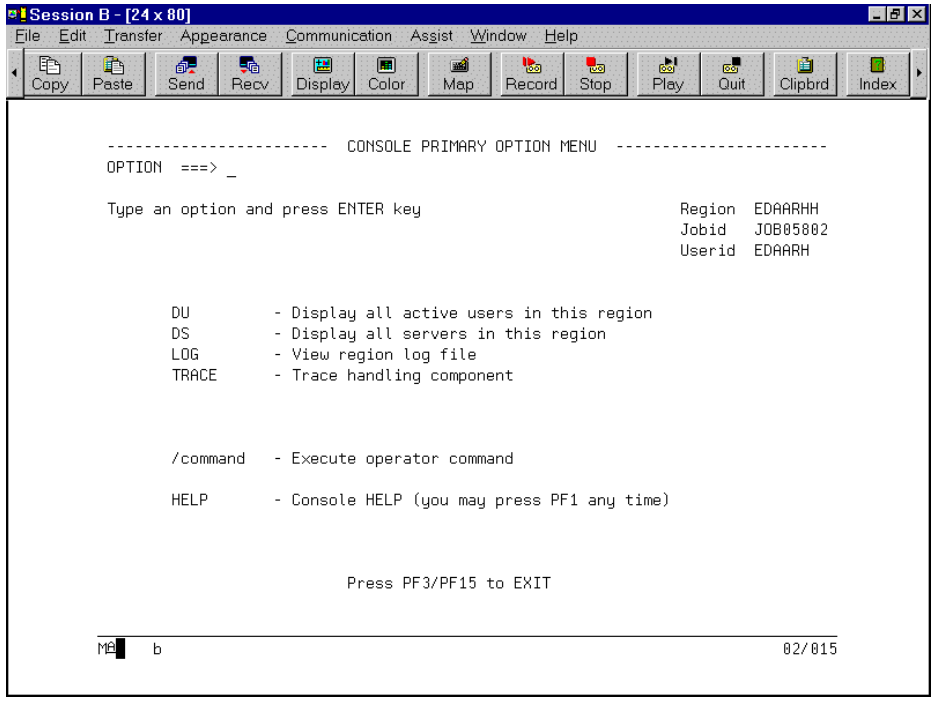
2. You will see the region name (in this example, the region name is EDAARHF). If security is enabled on the server, you must enter a user ID and password into the fields provided.

Note: If CONSEC=INTERNAL and the server's security is off using EXTSEC=OFF in the EDASERVE file, then a password is *not* required.

3. Press the *Enter* key.

The console authenticates the user ID based on the security mode (INTERNAL or EXTERNAL) chosen in the server's configuration file (EDASERVE) at the time of configuration.

After a successful login, the Console Primary Option menu displays a list of console commands:



For details, see *Selecting Server Console Commands* on page 11-4.

Selecting Server Console Commands

Select command options from the Console Primary Option Menu, which opens when the logon process is complete. The console command options are:

Option	Description
DU	Displays the Console Display Users panel, which shows all active subtasks, the resource utilization for each subtask, and other details, such as priority and allocations. You need Operator privilege for most of the features under this option.
DS	Displays the Console Display Servers panel, which enables you to see the currently defined services, add new services, and dynamically change values defined by the server in the configuration file. In addition, you can quiesce, enable, and restart services.
LOG	Displays the Console Log Viewer, which shows the last two windows of output from EDAPRINT and contains system messages and server-generated error messages.
TRACE	Displays the Console Trace panel with currently active and deferred trace definitions.
/*command*	Enables you to execute server operator commands. For details, see *Executing MVS Operator Commands* on page 11-23.
HELP	Displays Help windows containing information on all commands and PF key usage. You can also get help from anywhere in the server console by pressing the PF1 key.

As you select certain main options such as DU (showing all active subtasks), you can use the following commands to navigate through the panels:

PF01/PF13	HELP	Use from anywhere in the console session.
PF03/PF15	EXIT	Exits you back to the Console Primary Option menu.
PF07/PF19	SCROLL UP	Scrolls you up through the current display.
PF08/PF20	SCROLL DOWN	Scrolls you down through the current display.
PF10/PF22	SCROLL LEFT	Scrolls you to the left of the current display.
PF11/PF23	SCROLL RIGHT	Scrolls you to the right of the current display.

Procedure How to Select Server Console Command Options

1. To choose an option, type the option name (DU, DS, LOG, TRACE, */command*, HELP) on the option command line at the top of the menu.

   For more information about these options, see *Selecting Server Console Commands* on page 11-4 and *Executing MVS Operator Commands* on page 11-23.

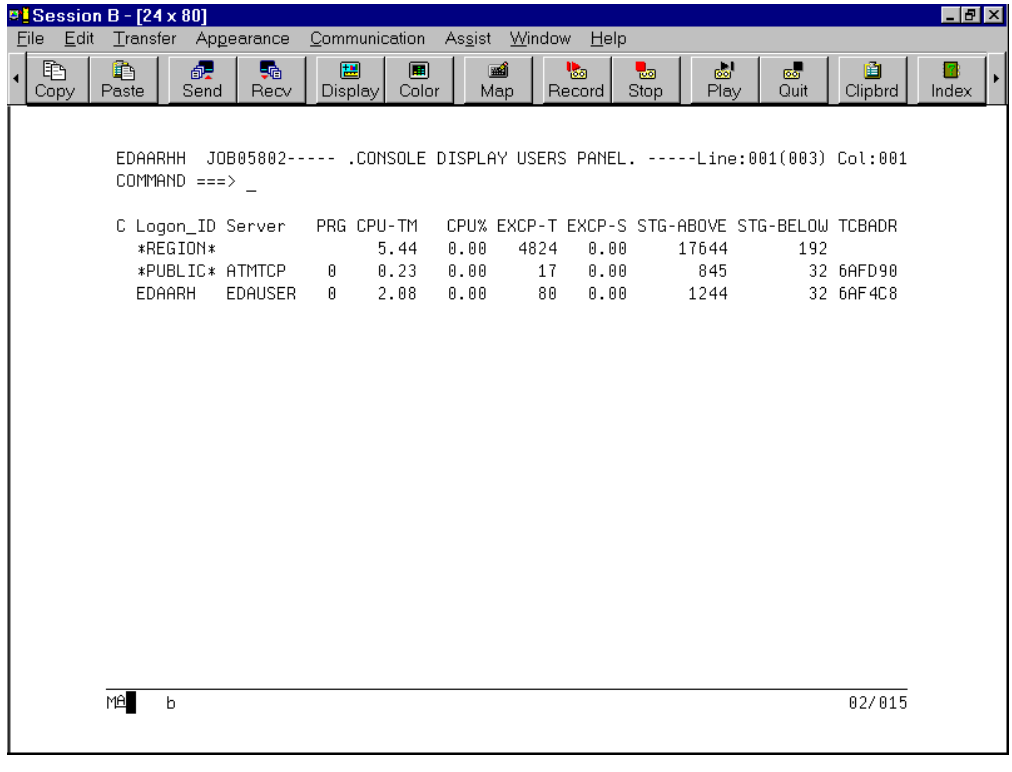
2. Press the *Enter* key.

Note: Any command on the command line without a "/" is considered to be a Console command.

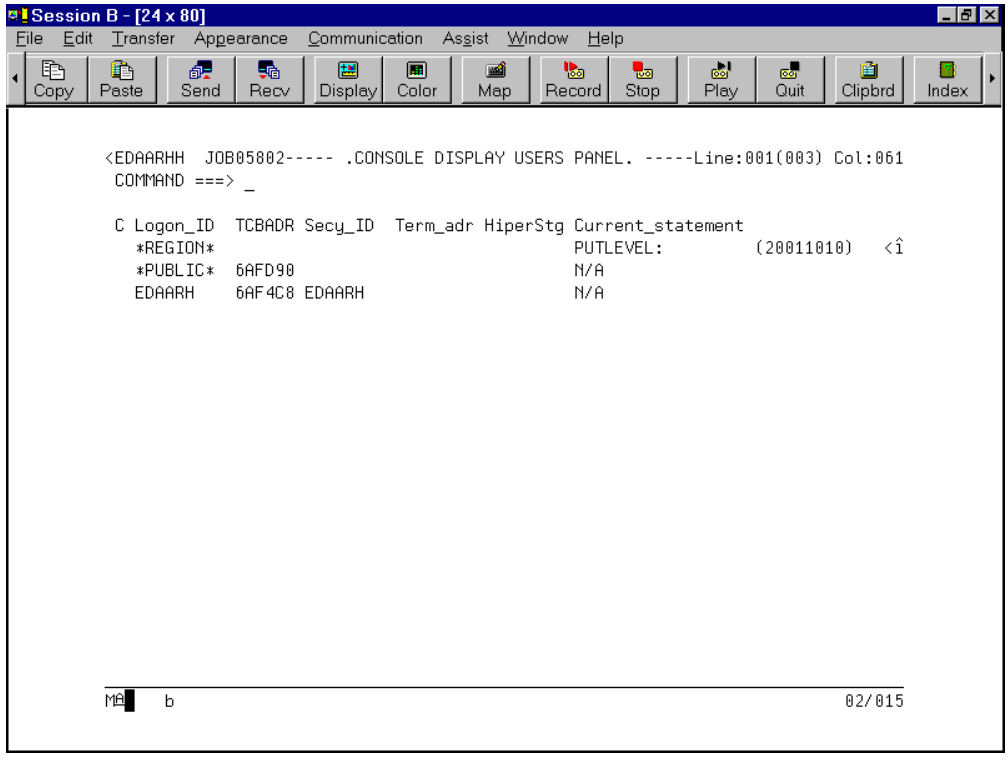
Procedure How to View the Console Display Users Panel

1. In the Console Primary Option menu, type *DU* at the command line to view the Console Display Users panel.

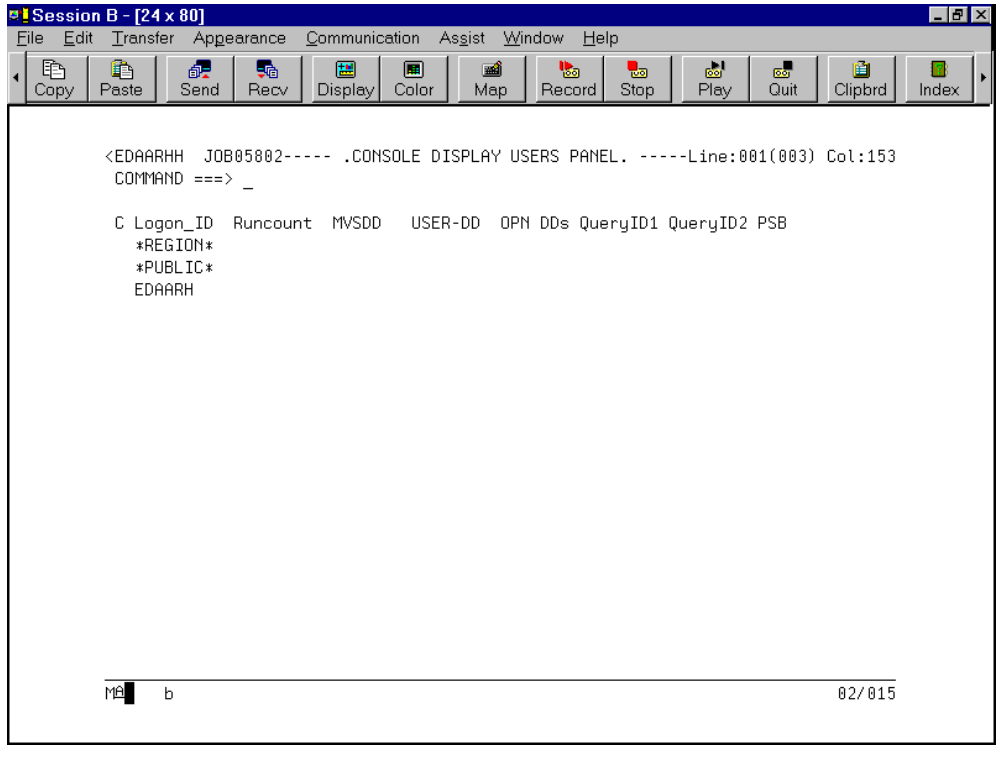
   The following are sample panels that display as you scroll from left to right. The first two columns are re-displayed from window to window for your convenience. The TCBADR column also repeats on the second window.



2. Scroll to the right (PF11) to view the continuation of the display:



3. Scroll to the right (PF11) to view the continuation of the display:



For details on each column of the Console Display Users panel, see *Console Display Users Panel* on page 11-8.

Reference Console Display Users Panel

Column Name	This column displays...

C	A one-character field (command column) in which an authorized user can issue operator commands. The following is a list of valid commands that you can issue in this column:
	`P`
	(Purge) Cancels a user or subtask.
	Note: A Purge will cancel the subtask and flush any associated output from the system.
	`C`
	(Cancel) Cancels a user or subtask.
	`W`
	Shows allocations for a specific data set name. All "W" prefix commands issued while in WHOHAS mode apply only to the data set criteria shown. If you are not in WHOHAS mode, or when you exit WHOHAS mode (using PF3), all subsequent "W" prefix commands refer to all data sets. Press the *Enter* key to update the display in real time. See *Working in WHOHAS Mode* on page 11-11.
	`S`
	(Select) Equivalent to W.
	`T`
	Switches to Trace mode for the selected user.
	Caution: If the P (Purge) or C (Cancel) command is entered next to the *REGION* line, it will bring the entire server down.

Logon_ID	The ID associated with the task. There are three valid types of IDs that display: • *REGION* represents the server region. • Service level. • Actual user ID.
Server	The service associated with each subtask. The service names that display in this column derive from the names designated in the active service blocks of the server configuration file (EDASERVE).

Column Name	This column displays...
PRG	The priority group of the current subtask. Valid values for the priority group range from 0 to 15 (where 15 is the highest priority). If the value is set to 0, a subtask will be denied CPU time. This value can be set by an authorized user. **Note:** This value does not change the MVS dispatching priority of the server region.
CPU-TM	The total amount of CPU time (in seconds) each task uses.
CPU%	The percentage of CPU time calculated from one occurrence of pressing the Enter key to the next time the Enter key is pressed. For *REGION*, the CPU% is the CPU time versus the total system CPU time. For USER, the CPU% is the percent of the USER's utilization versus the entire server's utilization.
EXCP-T	The total number of EXCPs initiated by a subtask or region during a given session. Certain EXCPs, such as module load EXCPs, are not included in this number.
EXCP-S	The average number of EXCPs per second that the subtask or region initiated since the last window refresh.
STG-ABOVE	The current amount of virtual storage utilized above the 16M line for the subtask or region. The value is presented in kilobytes (1,024 bytes).
STG-BELOW	The current amount of virtual storage utilized below the 16M line for the subtask or region. The value is presented in kilobytes (1,024 bytes).
TCBADR	The MVS Task Control Block (TCB) address of the given subtask in the server region.
Secy_ID	The eight-character ID the server uses as the System Authorization Facility (SAF) security ID. The site security package uses this ID to identify the user.
Term_adr	The terminal address of the active user logged on to the server. **Note:** This value only displays for LU2 users.
HiperStg	Reserved for future use.

Column Name	This column displays...
Current_statement	The last statement executed by the current user for all private LU2 services. This field has three components: • The remote procedure containing the statement. • The line number in the remote procedure. • The first characters of the line itself. If Logon_ID is equal to *REGION*, then this column displays the release level and the creation date of the server software.
Runcount	The number of attempts to connect to the central PDM. Runcount corresponds to the RUNCOUNT specification in the service block.
MVSDD	The real, translated, ddname used to allocate the file. If the file is allocated more than once by one user, this column will contain *MORE..., and the "W" command must be entered in the Command column to display detail on the allocations.
USER_DD	The untranslated ddname the user specified to allocate the file.
OPN	The number of times a data set referenced by a WHOHAS command is opened for a given task. See *Working in WHOHAS Mode* on page 11-11.
DDs	The number of matches in a subtask for a data set referenced by a WHOHAS command. See *Working in WHOHAS Mode* on page 11-11.
Query IDI	Reserved for future use.
Query ID2	Reserved for future use.
PSB	The name of the PSB being accessed (for IMS access only).

Reference Working in WHOHAS Mode

All W prefix commands issued while in WHOHAS mode apply to only the data set criteria shown. When you exit WHOHAS mode (using PF3), all subsequent W prefix commands refer to all data sets.

If the requested data set is allocated to an active task in the region, and if this active task has a *single* allocation, then the console displays:

- The ddname utilized by MVS in the MVSDD column.
- The ddname utilized in the allocation of the data set in the USER-DD column.
- The number of times that the data set has been opened in the OPN column.

If the requested data set is allocated to an active task in the region, and if the active task has *multiple* allocations, the console displays:

- *MORE... in the MVSDD column.
- The number of times that the file has been allocated in the DDs column.

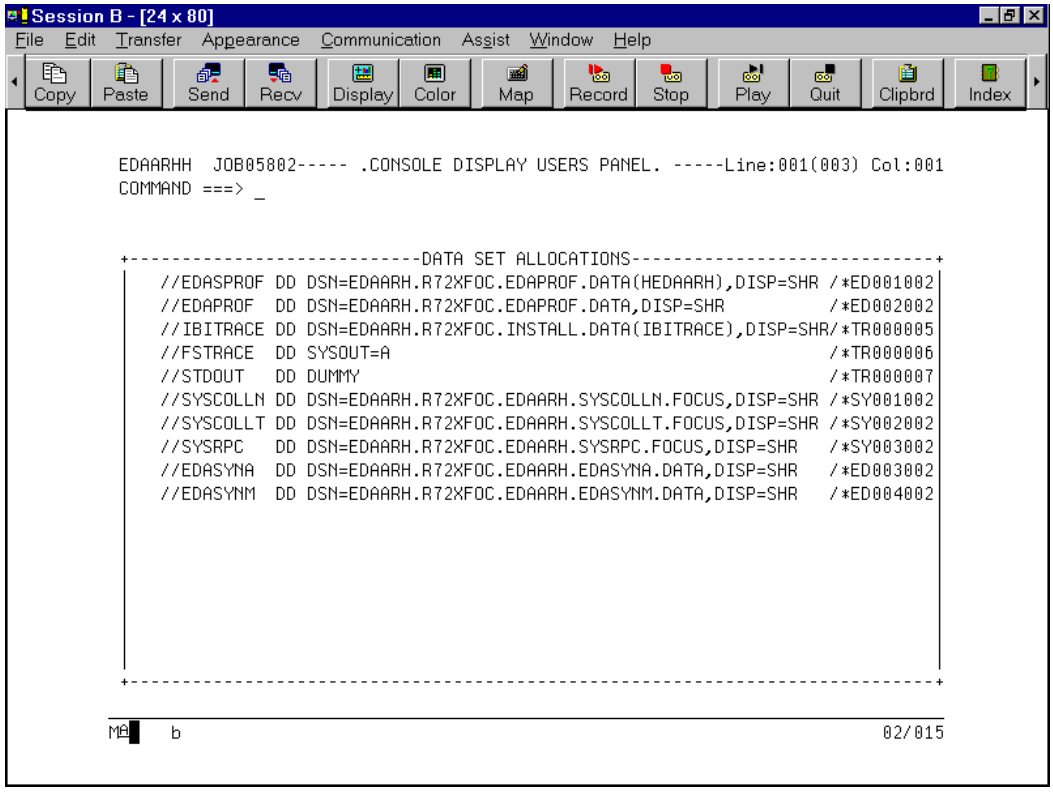
To view detail information for multiple ddnames, place a W next to the line that contains *MORE.... This will display a new window for Data Set Allocations.

If the OPN count is:

- A non-zero, then MVS cannot allow that allocation to be freed. Terminating the subtask frees this data set, but can also cause all current processing in that subtask to be terminated.
- Zero, then the DYNAM command can be utilized to free the data set. For more information on the DYNAM command, see the *iWay Stored Procedures Reference* manual.

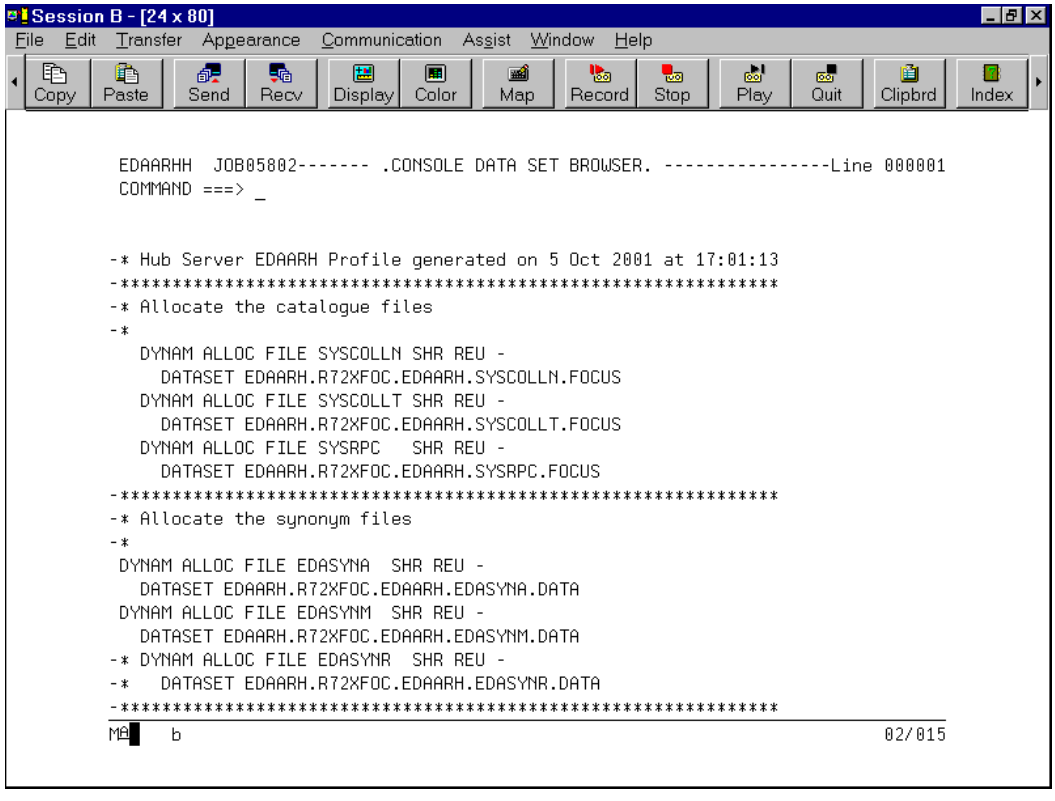
Example Viewing the Output From an S (Select) Command

The following is an example of the output from an S (Select) command for user EDAARHF. The output shows files allocated in the SERVINIT file and profiles. Note that both ddnames and translated ddnames are displayed.



An additional S in column 1 enables you to browse an individual data set.

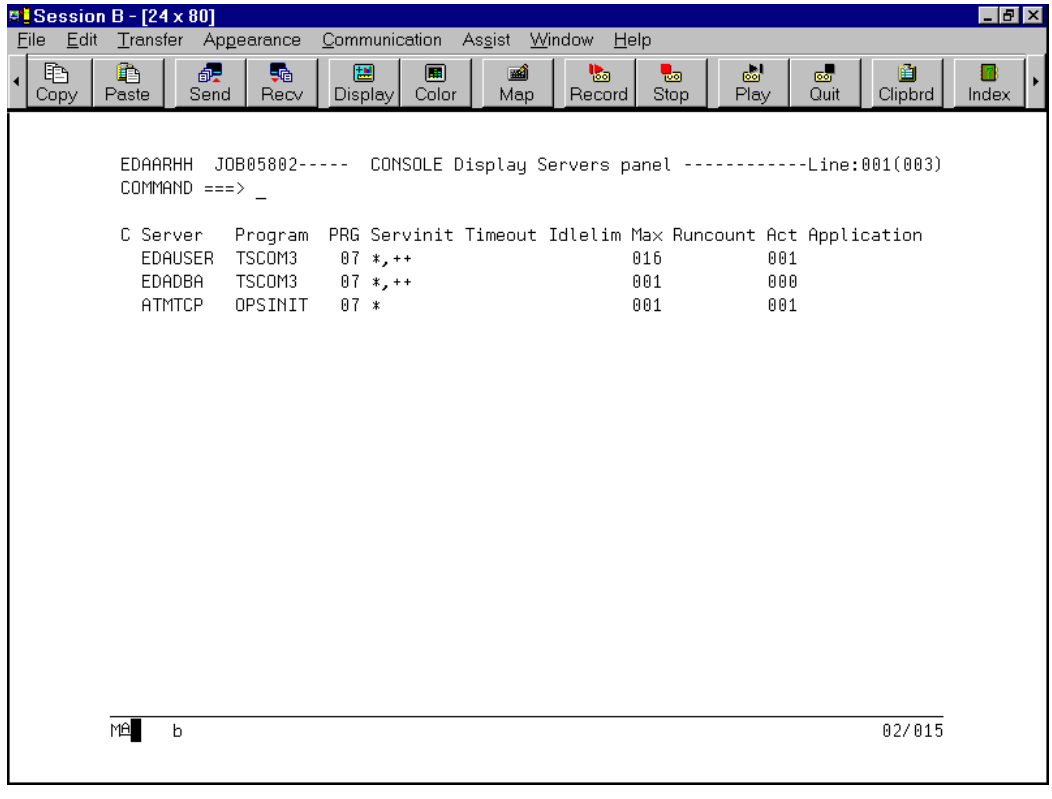
The following output is a sample result of IBITRACE. While displaying a data set or data set member, the command HEX ON appears in hexadecimal mode and HEX OFF appears in normal character mode.



Procedure How to View the Console Display Servers Panel (DS)

From the Console Primary Option Menu, type *DS* to view the Console Display Servers panel.

This panel enables a console user to see new services present in the server address space. It also enables an authorized user to control the services in the region. Several values from the server configuration file can be altered.



For details on each column of the Console Display Servers panel, see *Console Display Servers Panel* on page 11-15.

Reference Console Display Servers Panel

Column Name	This column displays...

C	A one-character field (command column) in which an authorized user can issue operator commands. The following is a list of valid commands that you can issue in this column:
	`Q`
	(Quiesce Service) This command enables an authorized user to prohibit new connector logons to a private service.
	`E`
	(Enable Service) This command enables an authorized user to allow a quiesced service to accept new logons.
	`I`
	(Insert Service) This command enables an authorized user to add a new service to the server environment. After performing the insert, you must supply the parameters associated with the service by typing over the appropriate columns on the display. This new service appears at the bottom of the list. You must supply the service characteristics.
	`S`
	(Start Service) This command enables an authorized user to start a server task using the values defined on this service definition.
	`D`
	(Delete Service) This command enables an authorized user to delete a new service from the server environment.
	`T`
	Switches to Trace mode for the selected server.

Server	The name of the service as defined in the server configuration file (EDASERVE). You can change this value by typing over the specific value on the display window. **Caution:** By changing the name of the service you can prevent users from accessing it. Since all connect requests specify a service name, changes to service names must be communicated to the appropriate users and/or application developers at your site.

Column Name	This column displays...
Program	The name of the program that the service executes. This value is derived from the PROGRAM parameter for each service defined in the server configuration file. You can change the value for this parameter by typing over the specific value on the display window. **Caution:** This may change the behavior of the service, causing negative impact on applications or users.
PRG	The priority group of the service. The values range from 0 to 15. If the PRTYGROUP parameter is not specified in the server configuration file, the default value of 7 is used. If the value is 0, then a service will be denied CPU time, effectively halting the service and hanging all users connected to it.
Servinit	The Service Initialization file the service uses (and the SERVINIT parameter) in the server configuration file.
Timeout	The time in seconds that a privately deployed service, which utilizes the LU2 protocol, can be in a terminal wait state before it is terminated. **Note:** It is recommended that you use the IDLELIM parameter in the server configuration file to provide this function.
Idlelim	The time in CPU seconds that a task can be idle before it is terminated.
Max	The maximum number of users that can initiate this type of service. The value displayed comes from the value specified for MAXIMUM in the server configuration file.
Runcount	Specifies the number of attempts to connect to the central PDM. Runcount corresponds to the “RUNCOUNT” specification in the service block.

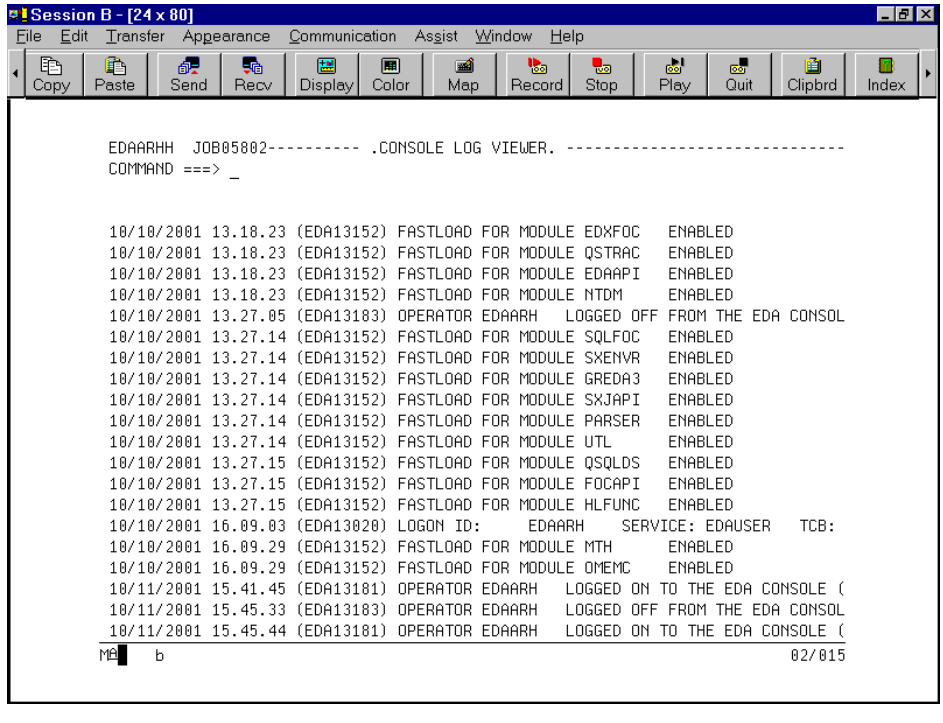
Column Name	This column displays...
Act	The actual number of active services for processing requests. The actual number cannot exceed the maximum for that service. **Note:** When a service ends in an active server, a new service is initialized. Therefore, the number of actual services does not decrease when a user logs off. This number only decreases when a service ends on a region or subtask that is quiesced, or TASK_RESTART is set to OFF in the EDASERVE configuration file.
Application	Reserved for future use.

Procedure How to View the Console Region Log Panel (LOG)

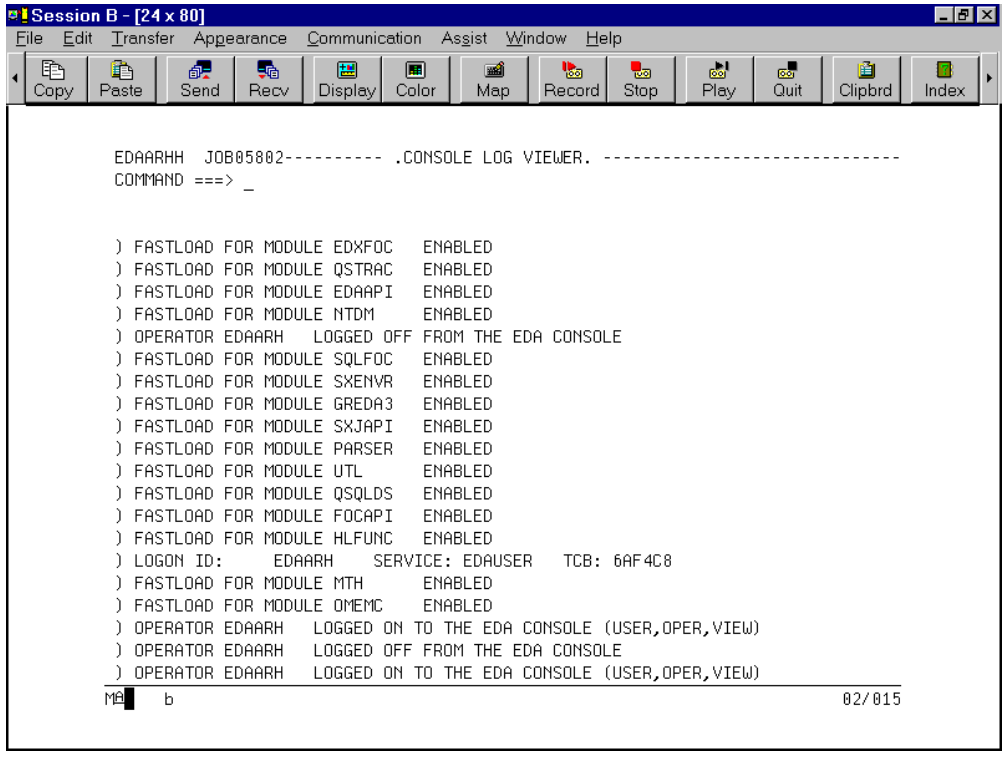
This option enables an authorized user to view the last two windows of the EDAPRINT log in a server region.

1. In the Console Primary Option menu, type *LOG* at the command line to view the Console Log Viewer window.

   You can use PF7/PF8 to scroll up and down and use PF10/PF11 to scroll left and right.

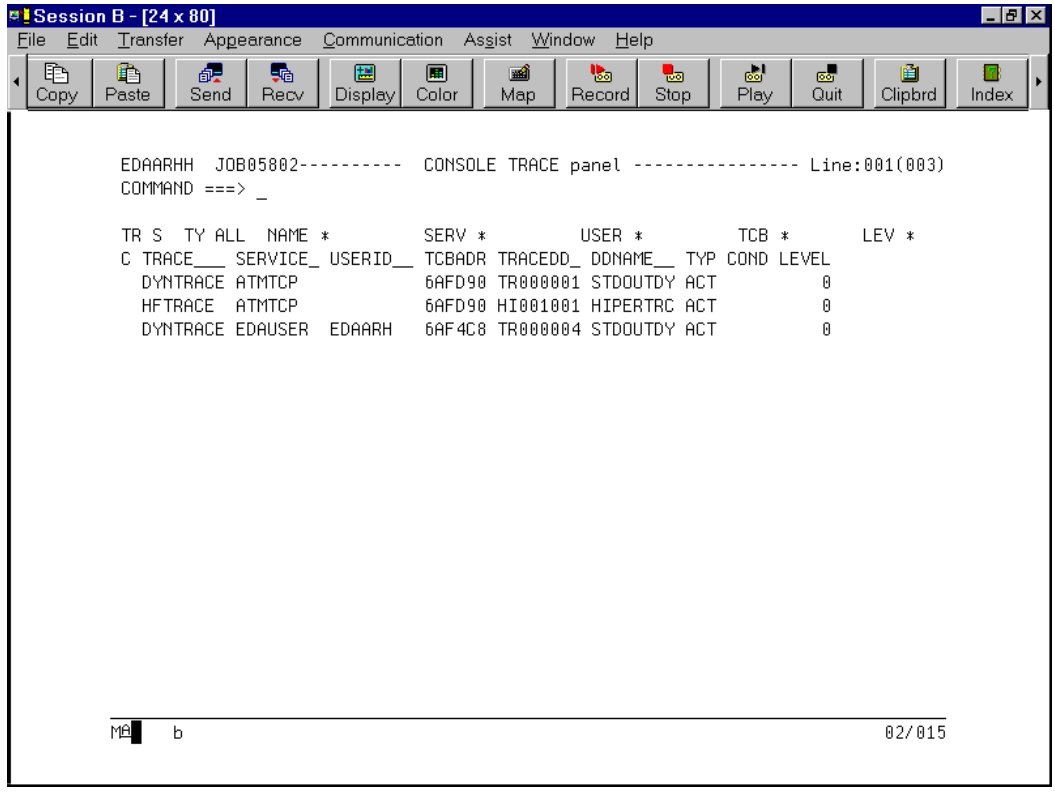


2. Scroll to the right (PF11) to view the continuation of the display:



Procedure How to View the Console Trace Panel

In the Console Primary Option menu, type *TRACE* at the command line to view the Console Trace panel:



For details on each column of the Console Trace panel, see *Console Trace Panel* on page 11-20.

Reference Console Trace Panel

Column Name	This column displays...
C	A one-character field (command column) in which an authorized user can issue operator commands. The following is a list of valid commands that you can issue in this column: S (Start Trace) This command enables an authorized user to start a trace. It switches the trace level to the default. P (Stop Trace) This command enables an authorized user to stop a trace. It switches the trace level to zero.
TRACE	The name of the trace.
SERVICE	The service name under which the task is running.
USERID	The user ID associated with the task.
TCBADR	The MVS Task Control Block (TCB) address of the given subtask in the server region.
TRACEDD	The real (translated) ddname to which trace records are written.
DDNAME	The untranslated ddname, which is used for writing trace records.
TYP	The type of the trace. It contains one of the following values: ACT An active trace is a trace that is currently running. It only produces output if it is set to an appropriate level. DEF A deferred trace is a trace that is not currently running. It is waiting for a trace matching its selection criteria to start.
COND	The value COND if the trace is a conditional trace.
LEVEL	The trace level, which is switched to the default level when the trace is started. It switches to level zero when the trace is stopped.

Searching for Allocations

You may need to know who is currently using a data set allocated under a specific server region. You can accomplish this by searching for allocations using any of the Console Display Users panels. For details, see *Console Display Users Panel* on page 11-8.

Syntax How to Search for All or Specific Allocations

You can quickly search for all or specific allocations by issuing the WHOHAS command

```
WHOHAS dsn
```

where:

dsn

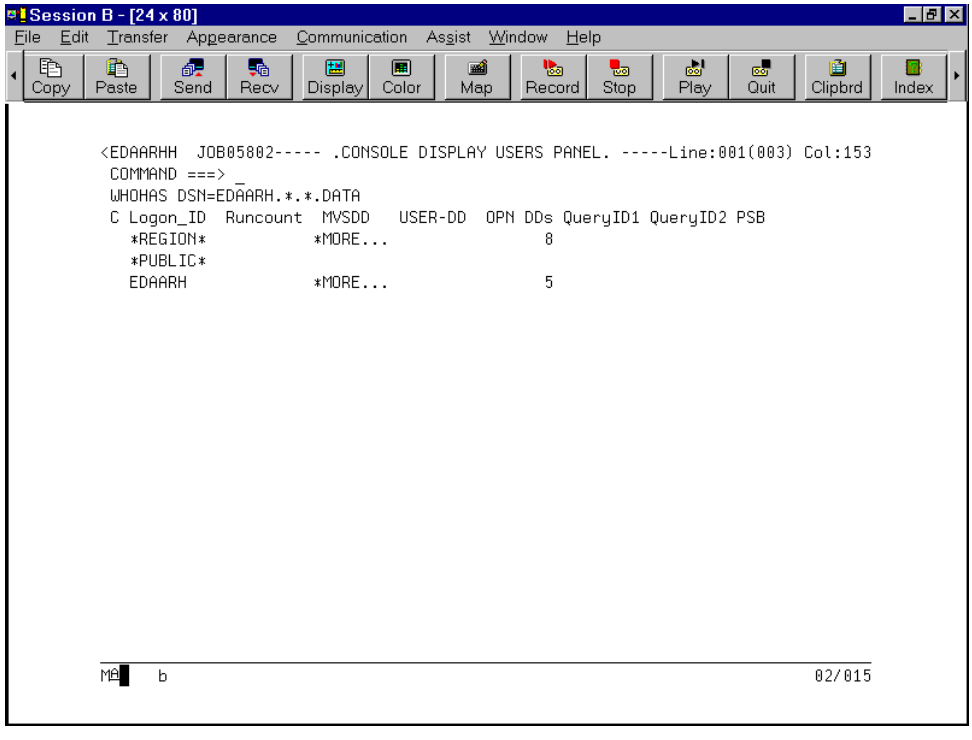
Can be a partial, full, or all data set names. You can specify partial data set names with the wildcard character, asterisk (*), which indicates zero-to-many positions. An asterisk can also be used to indicate all data sets. For details, see *Working in WHOHAS Mode* on page 11-11. For an illustration, see *Searching for Specific Allocations* on page 11-22.

Example Searching for Specific Allocations

The following is an example of searching for a specific allocation:

```
WHOHAS SYS1.*.PROCLIB
```

When you issue this command from the Console Display Users panel, the window automatically scrolls to the right and opens:



The console is then in WHOHAS mode, indicated by the WHOHAS line, which appears below the command line and above the DU display. For related information, see *Working in WHOHAS Mode* on page 11-11.

Executing MVS Operator Commands

You can issue MVS operator commands from the server console or from the MVS operating system, with some syntax variations.

From the...	Syntax
Server Console	`/cmd parameters`

MVS Operating System	`/F jobname, cmd parameters`
	or
	`/F server_name, cmd parameters`

where:

`/`

Is required syntax.

`cmd`

Is the MVS operator command. For a complete list, see *MVS Operator Commands* on page 11-24.

`parameters`

Are the settings available for the specific command.

`jobname`

Is the server job name or started task name. This syntax, preceded by F, is required for commands issued directly from MVS.

`server_name`

Is the name of the adapter. This syntax, preceded by F, is required for commands issued directly from MVS.

Reference MVS Operator Commands

The following is a list of available MVS operator commands:

Command	Description
CAN[CEL]	Forces disconnection of a user. For details, see *Terminating Connector Connections* on page 11-25.
CHAP	Changes the priority group of the specified MVS subtask. For details, see *Changing the Priority of the Subtask* on page 11-27.
DBCTL (and associated commands)	Starts and stops the DBCTL thread, modifies the security class, and changes the DFSPZPxx module. For details, see *Controlling the DBCTL Environment* on page 11-28.
DISPLAY	Displays statistics. For details, see *How to Display Statistics Based on the Configuration File Specifications* on page 11-33.
DUMP	Sets the dump support characteristics provided by the Scheduler or Initiator. For details, see *Setting Dump Support Characteristics* on page 11-29.
ENABLE	Allows new users to connect by removing the restrictions imposed by the QUIESCE command. For details, see *Enabling Connector Connections* on page 11-31.
MN_INTERVAL	Activates monitoring in intervals. For details, see *How to Monitor in Intervals* on page 11-34.
MN_REGION	Activates monitoring to include region statistics. For details, see *How to Monitor Region Statistics* on page 11-35.
MN_STORAGE	Activates monitoring to include storage statistics. For details, see *How to Monitor Storage Statistics* on page 11-35.
MN_USERS	Monitors statistics by user ID. For details, see *How to Monitor by User ID* on page 11-36.
MONITOR	Activates monitoring. For details, see *How to Control Monitoring* on page 11-34.
ON DEMAND	Requests On Demand statistics. For details, see *How to Request On Demand Statistics* on page 11-33.

Command	Description
QUIESCE	Prevents new subtasks from being started. For details, see *Preventing New Connections* on page 11-36.
SHUTDOWN	Quiesces all services by terminating all subtasks and freeing all resources upon completion of actual processing. For details, see *How to Terminate All Agent Processes (SHUTDOWN)* on page 11-26.
TRACE	Activates and manages various traces. For details, see *MVS Job Tracing* on page 11-37.
USERS	Displays a list of all subtasks on the server and their respective statistics. For details, see *How to Display All Active Server Subtasks* on page 11-38.

Terminating Connector Connections

There are two methods of terminating connector connections. You can either disconnect an individual user or shutdown the entire server.

Syntax

How to Terminate Individual Connector Connections

The CANCEL command forces disconnection. You can cancel a user connected to the server, and currently occupying a server subtask, by issuing the CANCEL command. CANCEL, or CAN, is used to disconnect a specific user by purging his or her security ID, logon ID, or TCB address. All duplicate security IDs or logon IDs will also be canceled. Any processing done by the canceled IDs terminates immediately and the resources utilized by the IDs release.

Note: The CANCEL function is currently available for all private services only.

Issue the following syntax for the CANCEL command from the server console or from MVS

```
/[F jobname, ]{CANCEL|CAN} {idname}
```

where:

jobname

Is the server job name or started task name.

idname

Is a security ID, logon ID, or TCB address. Possible values are:

SECID|SID=*security ID* is the security ID assigned to the user by the security subsystem. Typically, this ID is the logon ID.

LOGONID|LID=*logon ID* is the logon ID used by the connector or application to sign on to the server.

TCB=*tcb address* is the address of the user's TCB.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Example Using the CANCEL LOGONID Command

To cancel all users with the logon ID THISUSR who are logged on to the server, EDASERV, in any private service, issue the following:

```
/[F EDASERV, ]CANCEL LOGONID=THISUSR
```

Note: If an asterisk (*) is used in place of a security ID or logon ID, then all users are immediately canceled.

Example Using the CAN SECID Command

To cancel all users logged on to the server, EDASERV, issue the following:

```
/[F EDASERV, ]CAN SECID=*
```

You can also issue a force disconnect from the DU panel by entering a C (Cancel) command or a P (Purge) command. A purge command cancels the user's connection and flushes any associated output from the system.

Syntax How to Terminate All Agent Processes (SHUTDOWN)

The SHUTDOWN command (when issued without NOW and CANCEL options) quiesces all services, terminating all subtasks and freeing all resources upon completion of actual processing. The server for MVS should be terminated with a normal shutdown whenever possible, to ensure release of cross address space resources.

Issue the following syntax for the SHUTDOWN command from the server console or from MVS

```
/[F jobname, ]SHUTDOWN {NOW|CANCEL}
```

where:

jobname

Is the server job name or started task name.

NOW

Is equivalent to a STOP command.

CANCEL

Reverses a pending SHUTDOWN.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Changing the Priority of the Subtask

The Change Priority (CHAP) command changes the Server Resource Manager (MRM) priority group of the specified MVS subtask.

Syntax How to Change the Priority of the MVS Subtask

Issue the following syntax to change the priority group from the server console or from MVS

```
/[F jobname, ]CHAP TCB=tcb address, priority_level
```

where:

jobname

Is the server job name or started task name.

tcb address

Is the address of the user's TCB.

priority_level

Must be a value ranging from 0 through 15. The highest priority is 15; the lowest priority is 1. If 0 is specified, then the subtask suspends. You must subsequently set a subtask from 0 back to a higher value if you wish it to resume execution.

This command is equivalent to overtyping the value of the PRG column in the DU panel.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Controlling the DBCTL Environment

You can start and stop the DBCTL thread, modify the security class, and change the DFSPZP*xx* module that is loaded at initialization time.

Syntax How to Start and Stop the DBCTL Thread

To start and stop the DBCTL thread, issue the following commands from the server console or from MVS:

```
/[F jobname, ]IMSPZP=dra_name

/[F jobname, ]DBCTL={START|STOP[,I]}

/[F jobname, ]IMSSEC={ON|OFF}

/[F jobname, ]IMSCLASS=class_name
```

where:

jobname

Is the server job name or started task name.

dra_name

Sets the name of the DRA Startup Table. Must be issued prior to the /DBCTL=START command.

START

Starts the DBCTL connection to IMS.

STOP

Stops the DBCTL connection to IMS after waiting for all currently executing requests to finish.

STOP,I

Stops the DBCTL connection to IMS immediately; does not wait for requests to finish.

ON

Enables security checking.

OFF

Disables security checking.

class_name

Is a valid class that contains security authorization information.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Setting Dump Support Characteristics

The DUMP command enables you to set the dump support characteristics provided by the Scheduler or Initiator. It enables you to activate or deactivate dump support and to customize the output characteristics of the dump.

Procedure How to Configure Dump Support

To enable this feature you must either:

Allocate EDADUMP in the server JCL as follows:

```
//EDADUMP     DD SYSOUT=*
```

or

Allocate EDADUMP using the DYNAM command from the console. For more information on the DYNAM command, see the *iWay Stored Procedures Reference* manual.

If this allocation is present but dump support is not required, you can disable it by placing the following command in the global section of the server configuration file (EDASERVE):

```
DUMP=OFF
```

Syntax How to Control Dump Support

Issue the following syntax for the DUMP command from the server console or from MVS

```
/[F jobname, ]DUMP {ON|OFF}, {DUMP|TRACE}, {NOMVS|MVS}, {USER|NOUSER},
    {CTL|NOCTL}, {LONG|SHORT}, {REQ|ANY}
/[F jobname, ]DUMP TCB=tcb address
/[F jobname, ]DUMP ?
```

where:

jobname

Is the server job name or started task name.

ON

Activates dump support. YES is a synonym for ON.

OFF

Deactivates dump support. NO is a synonym for OFF. Note that deactivating the dump support for the Scheduler or Initiator does not affect the ability of the MVS operating system to perform dumps.

DUMP

Indicates that MVS-type control blocks are included in the dump, as well as the Scheduler or Initiator internal traces. This value is the default.

`TRACE`

Indicates that only the Scheduler or Initiator internal traces are included in the dump.

`NOMVS`

Indicates that only the Scheduler/Initiator dump is to be generated. This value is the default.

`MVS`

Indicates that MVS dumps are made in addition to the Scheduler or Initiator dump. Therefore, if a SYSUDUMP, SYSABEND, or SYSMDUMP ddname is included in your Scheduler or Initiator JCL, MVS generates a SYSUDUMP, SYSABEND, or SYSMDUMP dump as well.

`USER`

Indicates that all server subtasks generate a Scheduler or Initiator dump when they abend. This value is the default.

`NOUSER`

Indicates that server tasks do not generate a dump when they abend.

`CTL`

Indicates that the control tasks generate the Scheduler or Initiator dump when they abend. This value is the default.

`NOCTL`

Indicates that the control tasks do not generate the Scheduler or Indicator dump.

`LONG`

Indicates that the long form of the Scheduler or Initiator dump is generated. This value is the default.

`SHORT`

Indicates that the abbreviated form is generated.

`REQ`

Requests the Scheduler or Initiator to suppress abends and not to create a dump. This value is the default.

`ANY`

Requests that the Scheduler or Initiator generates a dump even if MVS does not request one.

`tcb address`

Is the TCB address of the server subtask. It dumps the task associated with the TCB address.

?

Displays the current Scheduler or Initiator dump support characteristics.

The Scheduler or Initiator dump support allows the operator to dump a single server subtask, as well as cause dumps to be automatically generated whenever a user or a control task abends.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Syntax How to Set the Scheduler or Initiator Dump Support

To set the Scheduler or Initiator dump support characteristics, issue the following:

```
/[F EDASERV, ]DUMP ON,NOMVS,CTL,LONG,DUMP
```

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Syntax How to Dump the Task Identified by the TCB Address

To dump the task identified by the TCB address 8C7C90, issue the following:

```
/[F EDASERV, ]DUMP TCB=8C7C90
```

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Enabling Connector Connections

You can reactivate the server or service after a QUIESCE command by issuing an ENABLE command. The ENABLE command allows connector applications to connect to the server.

Syntax How to Reinstate Connector Connections

The ENABLE command allows new users to connect, removing the restrictions imposed by the QUIESCE command.

To reinstate the ability for new users to connect to the service after a QUIESCE command, you must issue the ENABLE command.

Issue the following syntax for the ENABLE command from the server console or from MVS

```
/[F jobname, ]ENABLE [service]
```

where:

jobname

Is the server job name or started task name.

service

Is the service that will resume processing. When service is omitted, all services are enabled.

This command is equivalent to typing an E in the command column in the Console Display Servers panel.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Example Using the ENABLE Command

To allow new users to connect to any service on the server, EDASERV, issue the following from the server console or from MVS:

```
/[F EDASERV, ]ENABLE
```

Server Monitoring and Statistics

Server monitoring and statistics provide detailed server resource utilization reports for a region or for all users. The information can be viewed immediately online or evaluated in EDAPRINT. All output defaults to EDAPRINT, except Short on Storage messages, which are also sent as non-scrollable messages to the operator console (WTO). The extent of information collected is determined by the specific command set issued from the server console or configuration file. Shutdown statistics are always produced and cannot be disabled. Any command can be turned on or off using the console. For details, see *Resource Utilization Reports* on page 11-33.

Reference Resource Utilization Reports

The following table describes the resource utilization reports provided by server monitoring and statistics:

On Demand	Reports server resource statistics for users on an on demand basis.
Monitoring	Reports server resource statistics for users at user specified intervals.
Short on Shortage	Displays a console message when a region reaches a site-defined threshold.
Shutdown	Reports summary statistics by service at termination for the server region.

Syntax How to Request On Demand Statistics

On Demand statistics are produced whenever requested. To request On Demand statistics, issue the following syntax from the server console:

```
ON DEMAND
```

Syntax How to Display Statistics Based on the Configuration File Specifications

To display statistics based on the configuration file specifications, issue the following command from the server console or from MVS

```
/[F server_name, ]DISPLAY STATS[,ALL]
```

where:

server_name

Is the name of the server.

ALL

Displays all statistics regardless of the configuration file specifications.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Syntax How to Control Monitoring

Monitoring statistics are produced at user specified intervals. (APF authorization is required.)

To turn monitoring on or off, issue the following command from the server console or from MVS

```
/[F server_name, ]MONITOR {OFF|ON}
```

where:

server_name

Is the name of the server.

OFF

Resets the current setting for MN_USER and MN_INTERVAL to the original values in the configuration file. The default value is OFF.

ON

Initiates monitoring.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Syntax How to Monitor in Intervals

To activate monitoring to perform at integer multiples of the MRM_INTERVAL (MVS Resource Manager Interval), issue the following command from the server console or from MVS after monitoring is initiated

```
/[F server_name, ]MN_INTERVAL=monitor_interval
```

where:

server_name

Is the name of the server.

monitor_interval

Specifies the interval at which to perform monitoring (when active). The value is an integer indicating the interval in seconds. The minimum is 30 and the maximum is 3,600. The default value is 1,800 (30 minutes).

APF authorization is required.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Syntax

How to Monitor Storage Statistics

To activate monitoring to include storage statistics, issue the following command from the server console or from MVS after monitoring is initiated

```
/[F server_name, ]MN_STORAGE={OFF|ON}
```

where:

server_name

Is the name of the server.

OFF

Specifies monitoring not to include storage statistics. The default value is OFF.

ON

Specifies monitoring to include storage statistics.

APF authorization is required.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Syntax

How to Monitor Region Statistics

To activate monitoring to include region statistics, issue the following command from the server console or from MVS after monitoring is initiated

```
/[F server_name, ]MN_REGION={OFF|ON}
```

where:

server_name

Is the name of the server.

OFF

Specifies monitoring not to include region statistics. The default value is OFF.

ON

Specifies monitoring to include region statistics.

APF authorization is required.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Syntax How to Monitor by User ID

To monitor statistics by user ID, issue the following command from the server console or from MVS after monitoring is initiated

```
/[F server_name, ]MN_USERS={OFF|ALL}
```

where:

server_name

Is the name of the server.

OFF

Curtails monitoring of user IDs. The default value is OFF.

ALL

Produces usage statistics for all logged on user IDs.

APF authorization is required.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Preventing New Connections

You can perform maintenance work on the server without terminating any connections by issuing a QUIESCE command. The QUIESCE command prevents new connector applications from connecting while allowing existing applications to finish processing. You may quiesce some or all services in the server.

Once the existing users have terminated their sessions, you can perform any maintenance work and then reactivate the server or service.

Syntax How to Quiesce Server Services

The QUIESCE command prevents new subtasks from being started. You may quiesce some or all services in the server. Services will also reject new connect requests, while continuing to process for connected users.

Issue the following syntax for the QUIESCE command from the server console or from MVS

```
/[F jobname, ]QUIESCE [service]
```

where:

jobname

Is the server job name or started task name.

service

Is the name of a service defined using the configuration file (EDASERVE). When service is omitted, all services are quiesced.

This command is equivalent to typing a *Q* in the command column in the Console Display Servers panel.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

MVS Job Tracing

Tools to activate and manage the various traces are available to MVS system operators and users of the server console.

Syntax How to Activate and Manage Traces

The TRACE command allows you to activate and manage the various traces that may be generated.

Issue the following syntax for the TRACE command from the server console or from MVS

```
/[F jobname, ]TRACE trace_name spec_trace {ACTive|DEFer}
trace_restriction COND
```

```
/[F jobname, ]QUERY trace_name {FULL|SHORT}
```

where:

jobname

Is the server job name or started task name.

trace_name

Identifies the trace. An “*” or a “?” may be used as a wildcard.

spec_trace

Indicates whether to start or stop the trace, or specifies the trace level. Possible values are:

`START` starts the specified trace.

`STOP` stops the specified trace.

level specifies the level of the trace.

`ACTive`

Specifies that this command is to operate with currently active traces.

`DEFer`

Specifies that this command is to operate with traces that are not yet active.

trace_restriction

Restricts the scope of the trace command by screening on any combination of the following:

SERVICE=*service* screens on service name.

USERID=*security_id* screens on the security ID.

TCB=*tcb_address* screens on TCB address.

An "*" or a "?" may be used as a wildcard. The default value for all three keywords is "*". TCB is ignored for deferred traces.

COND

Defines whether or not the trace is conditional. It is only valid with the START option. A conditional trace is activated only if the appropriate ddname for the trace is previously allocated. The trace services will not dynamically allocate the trace file for a conditional trace.

QUERY

Displays information about the trace(s).

FULL

Displays all the trace information; it is only valid with the QUERY option.

SHORT

Displays a brief summary of the trace information; it is only valid with the QUERY option.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Syntax

How to Display All Active Server Subtasks

You can display all the active server subtasks in a server by using the USERS command.

The USERS command displays, on the MVS operations console, a list of all subtasks on the server. Issuing the USERS command also displays the CPU and EXCP statistics for each subtask. For PRIVATE subtasks, user information is provided. For public tasks, the user fields have the value PUBLIC displayed.

Issue the following syntax for the USERS command from the server console or from MVS

```
/[F jobname, ]USERS
```

where:

jobname

Is the server job name or started task name.

Note: Square brackets [ ] denote required syntax when issuing the command from the MVS operating system. To issue the command from the server console, omit the bracketed syntax.

Example

Using the USERS Command

To display all the subtasks on the server, EDASERV, and the values for the information associated with the subtasks, issue the following:

```
/[F EDASERV, ]USERS
```

The following information, which is associated with the subtasks, appears:

CONN	The service on which the user is active.
LOGONID	The user ID under which the user is logged on.
SECID	The security ID for the user.
FROM	The environment under which the user is operating his or her terminal.
TERM	The user's terminal ID.

For more information on usage accounting, see the *iWay Server Administration for MVS and VM* manual.

Index

A

ACB libraries 3-27

Access File libraries 3-51 to 3-52

access privileges for DB2 3-16

ACEE control blocks 8-5

ACF2 security system 8-6, 8-14

administration parameters for Resource Analyzer/ Governor 7-9

alternate code pages 5-12

APF authorization 3-22

APPLIDs for VTAM 2-15

Auto tools 7-16 to 7-17

B

Batch Autostart Facility 3-62

Batch Message Processing (BMP) 3-23 to 3-24, 3-26

BMP (Batch Message Processing) 3-23 to 3-24, 3-26

C

CA-ACF2 security system 8-6, 8-14

CALLIMS processing 5-29

CANCEL LOGONID command 11-26

case conversion 5-19

CASETBL file 5-19

CA-TOP SECRET security system 8-7, 8-14

central version access for CA-IDMS/DB 3-7

changing subtask priorities 11-27

CHAP command 11-27

CICS system 3-29

client connections 11-36

client node blocks 8-24

Cobol FD Translator 7-18

code page definition files 5-21, 5-23

code page generation list files 5-12

Collection Adapter parameter 7-9

Collection Name parameter 7-9

comments in configuration files 8-3

communication parameters for servers 2-8, 8-25, 8-30 to 8-31
- inbound LU6.2 2-8, 2-13, 8-25 to 8-26
- inbound TCP/IP 2-8, 2-12, 8-26, 8-29
- outbound LU6.2 2-10, 2-14, 8-32 to 8-33
- outbound TCP/IP 2-9, 2-13, 8-33, 8-36
- TCP/IP 8-35

communications configuration files 3-25, 3-31, 5-24, 8-20 to 8-21
- changing 5-24
- comments 8-3

communications gateways 1-5, 5-24
- configuring 5-25, 5-27
- security 5-28

communications subsystem configuration files 8-20 to 8-22

configuration files for servers 4-1 to 4-2
- Full-Function Servers 4-12, 4-14
- Hub Servers 4-4 to 4-5

configuration parameters for servers 2-6
- Full-Function Servers 2-11
- Hub Servers 2-6

Configuration Verification Progam (CVP) 7-14 to 7-15

configuring multiple server instances 3-2

connecting to a subserver 8-31

CONSEC parameter 8-10 to 8-12

Console Display Servers panel 11-14 to 11-15

Console Display Users panel 11-5, 11-8, 11-11 to 11-12

console for Server for MVS 8-8 to 8-14, 11-1 to 11-2
- command options 11-4 to 11-5
- panels 11-4 to 11-5
- starting 11-2

Console Log Viewer 11-17

Console Logon window 11-2, 11-5

Console parameters for servers 2-7, 2-11

Console Primary Option Menu 11-2, 11-5, 11-8, 11-14 to 11-15
- Console Trace panel 11-20

Console Trace panel 11-19 to 11-20

CPCODEPG file 5-10, 5-12

CPXCPTBL file 5-10, 5-12 to 5-13

CS/3 configuration files 8-20 to 8-21

CS3GWY files 5-25, 5-27

CS3SRV files 8-16

Customer Support Service 1-v
- How to contact 1-v
- Information required 1-vi

customizing monocasing tables for MVS 5-19

CV access for IDMS/SQL 3-23

CVP (Configuration Verification Progam) 7-14 to 7-15

D

Data Adapter for ADABAS 3-2

Data Adapter for CA-DATACOM/DB 3-3, 3-5

Data Adapter for CA-IDMS/DB 3-7 to 3-9, 3-12
- accessing 3-7 to 3-8

Data Adapter for DB2 3-14, 3-16 to 3-17, 3-20
- accessing 3-16
- security exits 3-21

Data Adapter for IDMS/SQL 3-21
- accessing 3-23
- APF authorization 3-22

Data Adapter for IMS 3-23

Data Adapter for IMS/DBCTL 3-34

Data Adapter for InfoMan 3-41, 3-46
- configuration files 3-42
- configuring 3-41
- enabling 3-43
- error processing 3-46

Data Adapter for Millennium 3-47
- customizing 3-47 to 3-48
- ddnames 3-54
- installing 3-51
- libraries 3-51
- requirements 3-47, 3-54

Data Adapter for MODEL 204 3-54

Data Adapter for Nomad 3-55

Data Adapter for Oracle 3-57

Data Adapter for SUPRA 3-58 to 3-59
- Batch Autostart Facility 3-62
- JCL code 3-62 to 3-63
- security 3-61

Data Adapter for SYSTEM 2000 3-64

Data Adapter for Teradata 3-65

data adapters 3-1 to 3-2
ADABAS 3-2
CA-DATACOM/DB 3-3, 3-7
CA-IDMS/DB 3-7
DB2 3-14
IDMS/SQL 3-21
IMS/BMP 3-23
InfoMan 3-41
Millennium 3-47
MODEL 204 3-54
Nomad 3-55
Oracle 3-57
SUPRA 3-58 to 3-59
SYSTEM 2000 3-64
Teradata 3-65

data set allocations 11-21
searching 11-21 to 11-22

DBCTL installation 3-38

DBCTL threads 11-28

ddnames for Data Adapter for Millennium 3-54

DEBUG parameter 3-59

DLI mode 3-27

DRA startup table 3-34
assembling 3-35
linking 3-35

DSN3SATH exit 3-21
for CA-CACF2 3-20
for CA-TOP SECRET 3-17
for RACF 3-17

DTCMURT1 job 3-4

DTCMURT2 job 3-5

DUMP command 11-29, 11-31

E

EDAAUTO tools 7-16 to 7-17

EDACFG configuration procedures 4-2

EDACFG procedures 8-2
member names 8-3

EDACSG communications configuration files 5-25, 5-27

EDAGATE communications gateway 1-5, 5-24
configuring 5-25, 5-27
security 5-28

EDALIB libraries 3-51

EDARPC commands 5-29

EDASERVE configuration files 4-2, 5-25, 5-27
security 5-28

ENABLE command 11-31 to 11-32

error processing for InfoMan 3-46

ERRORS libraries 3-51

ETL Servers 7-16

EXTSEC parameter 8-4, 8-10, 8-12

F

Fast Path databases 3-26

FILELIM parameter 5-8, 10-4

FOCBMP communications files 3-25, 3-31

FOCDSN3 program 3-17

FOCDSN4 program 3-17

FOCEXEC libraries 3-51

FOCPSB libraries 3-40

FOCSU data sets 7-2

FOCUS Database Server 7-2
configuring 7-2
Resource Governor and 7-12

Full-Function Servers 1-3 to 1-4
configuration files 4-12, 4-14
with Hub services 1-4
without Hub services 1-3

G

generating data adapters 3-1

generation list files 5-12

GKTABLE files 7-6, 7-11

GRANT command 7-13 to 7-14

granting access privileges 7-13

granting access privileges for DB2 3-16

H

HiperBUDGET feature 5-8, 10-4

HIPERCACHE parameter 5-6

HiperEDA feature 5-6, 10-4
- configuring 5-6
- requirements 5-6

HIPEREDA parameter 5-6

HIPEREXTENTS parameter 5-6

HIPERFILE parameter 5-6

HIPERINSTALL parameter 5-6

HIPERLOCKED parameter 5-6

HIPERSPACE parameter 5-6

hiperspace usage statistics 10-5

Hub Servers 1-2, 4-4 to 4-5
- configuration files 4-4 to 4-5

I

IBI Subsystem 5-2 to 5-3, 10-1, 10-3, 10-7
- benefits 5-2
- installing 5-3 to 5-4
- maintaining 5-5
- SUBSYSI program 10-2
- troubleshooting 10-6
- uninstalling 5-6

IGNORE parameter 3-46

IMANCONF configuration files 3-42

IMDTST program 3-38

IMS keywords 3-36

IMS transaction servers 5-28 to 5-29
- configuring 5-30 to 5-32
- requirements 5-30
- security 5-31

IMS/BMP environment 3-23 to 3-26

IMS/DBCTL environment 11-28
- configuring 8-15
- DRA startup table 3-34 to 3-35
- FOCPSB libraries 3-40
- installing DBCTL 3-38
- parameters for servers 2-12
- PSB resources 3-41
- security 3-40

inbound communication parameters for servers 8-25
- LU6.2 8-25 to 8-26
- TCP/IP 8-26, 8-29

inbound communications parameters for servers
- LU6.2 2-8, 2-13
- TCP/IP 2-8, 2-12

Installation Verification Program (IVP) 6-4 to 6-5

issuing MVS operator commands 11-23

IVP (Installation Verification Program) 6-4 to 6-5

J

JCL code 3-63

JCL for Data Adapter for DB2 3-14, 3-16

JCL for Data Adapter for InfoMan 3-43

K

keywords 3-36, 3-59
for IMS 3-36
for SUPRA multi-sessions 3-59

known code page files 5-12 to 5-13

known language list files 5-16 to 5-18

L

LANGTBL file 5-16 to 5-18

libraries for Data Adapter for Millennium 3-51

link-editing user-written programs 3-8 to 3-9, 3-12

linking Millennium API modules 3-52 to 3-53

local mode access for IDMS/SQL 3-8, 3-23

LU6.2 network definitions 2-15

LU6.2 parameters for servers 2-8
inbound 2-8, 2-13, 8-25 to 8-26
outbound 2-10, 2-14, 8-32 to 8-33

LU6.2 requirements for servers 2-3

LU6.2 service blocks 8-16, 8-18

LUPOOL ddname 4-12

M

MASTER libraries 3-51

member names 8-3

metadata generation 7-16 to 7-18

Millennium Access File libraries 3-51 to 3-52

Millennium API 3-47
linking modules 3-52 to 3-53

monocasing tables for MVS 5-19

multi-session keywords 3-59

MVS operator commands 11-23 to 11-24

MVS00 parameter 3-16

MVSLIM parameter 5-8, 10-4

N

National Language Support (NLS) 5-1
customizing monocasing tables 5-19
identifying enabled language names 5-16

National Language Support configuration files for MVS 5-9 to 5-10

NCRS numbers 3-16

network definitions 2-15 to 2-16

NLS (National Language Support) 5-19
customizing monocasing tables 5-19
identifying enabled language names 5-17

NLS configuration files for MVS 5-9 to 5-10

NLS sort tables 5-21

node blocks 8-24

NSTMTS numbers 3-16

O

On Demand statistics 11-33

operator commands for IBI Subsystem 10-3

outbound communication parameters for servers 8-30 to 8-31
LU6.2 8-32 to 8-33
TCP/IP 8-33, 8-36

outbound communication parameters for sub-server connections 8-31

outbound communications parameters for servers 2-9
LU6.2 2-10, 2-14
TCP/IP 2-9

outbound communications parameters for the Server for MVS 2-13

owner IDs 4-3

Owner Name parameter 7-9

P

parameters 5-6
- Collection Adapter 7-9
- Collection Name 7-9
- DEBUG 3-59
- EXTSEC 8-4, 8-10
- FILELIM 5-8, 10-4
- HIPERCACHE 5-6
- HIPEREDA 5-6
- HIPEREXTENTS 5-6
- HIPERFILE 5-6
- HIPERINSTALL 5-6
- HIPERLOCKED 5-6
- HIPERSPACE 5-6
- IGNORE 3-46
- IMS/DBCTL 2-12
- LU6.2 2-8, 2-10
- MVS00 3-16
- MVSLIM 5-8, 10-4
- Owner Name 7-9
- PICA 3-44 to 3-46
- PICACLSN 3-44 to 3-46
- PICADBID 3-44, 3-46
- PICASESS 3-44 to 3-46
- PICATINT 3-44 to 3-46
- PICAUSRN 3-44 to 3-46
- SECURITY 3-59
- Server Type 7-9
- TASK 3-59
- TCBLIM 5-8, 10-4
- TCP/IP 2-8 to 2-9, 2-13, 8-33
- THREADS 3-59

parameters for SET commands 5-6

PCB (Program Communication Blocks) 3-23, 3-32

PF keys for Server Console 11-4

PICA parameters 3-44 to 3-46

PICACLSN parameter 3-44 to 3-46

PICADBID parameter 3-44, 3-46

PICASESS parameter 3-44 to 3-46

PICATINT parameter 3-44 to 3-46

PICAUSRN parameter 3-44 to 3-46

preventing client connections 11-36

procedures 8-2
- EDACFG 8-2
- LU6.2 8-32

Program Communication Blocks (PCB) 3-23, 3-32

Program Specification Blocks (PSB) 3-32

protocol blocks 8-22
- TCP/IP 8-23

PSB (Program Specification Blocks) 3-32

PSB resources 3-41

PSBs (Program Specification Blocks) 3-40

Q

QUIESCE command 11-36

R

RACF security system 8-5, 8-13

RACROUTE macro 8-5

RE3JINS member 3-14, 3-16

RELJINS member 3-14, 3-16

Resource Analyzer 7-5 to 7-6
- administration parameters 7-9
- administrative data sets 7-11
- granting access privileges 7-13
- installing 7-9
- security 7-14
- simultaneous usage 7-6, 7-12
- trace files 7-16

Resource Governor 7-5 to 7-6
administration parameters 7-9
administrative data sets 7-11
FOCUS Database Server and 7-12
granting access privileges 7-13
installing 7-9
security 7-14
simultaneous usage 7-12
trace files 7-16
verifying configuration 7-14 to 7-15

S

SAF (System Authorization Facility) 8-4 to 8-5

searching for data set allocations 11-21 to 11-22

security exits for Data Adapter for DB2 3-17, 3-20 to 3-21

security for communications gateways 5-28

security for Data Adapter for SUPRA 3-61

security for IMS/DBCTL 3-40

security for Server for MVS 8-4 to 8-7

security for Transaction Server for IMS 5-31

SECURITY parameter 3-59

Select command 11-12

selection criteria in WHOHAS mode 11-11

server code pages 5-16

server console for MVS 8-8 to 8-14, 11-2
command options 11-4 to 11-5
panels 11-4 to 11-5
starting 11-2

Server for MVS 1-1, 2-2, 5-9, 6-4, 9-1
adding code pages 5-15
changing code page settings 5-11
client connections 11-25 to 11-26, 11-31 to 11-32
configuration requirements 2-2
configuring 1-1, 2-1, 2-6, 5-1, 8-1, 8-3
configuring for NLS 5-9
console 11-1
ending agent processes 11-26
identifying page settings 5-12
job tracing 11-37
monitoring 11-34 to 11-36
network definitions 2-15 to 2-16
optional components 6-7, 7-1
requirements 2-3
security 8-4 to 8-7
starting 9-2
start-up JCL 6-2
statistics 11-32 to 11-33
stopping 9-2
testing communications 6-1 to 6-2
upgrading 2-5 to 2-6
usage accounting 8-8

Server for MVS code pages 5-16
changing languages 5-17
changing settings 5-16
transcoding 5-16

server instances 3-2

server JCL 5-23

server monitoring 11-32 to 11-35
by User ID 11-36
in intervals 11-34

server security 8-4 to 8-7

Server Type parameter 7-9

server types 1-5
Full-Function 1-3
Full-Function with Hub services 1-4
Full-Function without Hub services 1-3
Hub 1-2

service blocks 8-16

SERVLIM parameter 5-8, 10-4

SET parameters 5-6
FILELIM 5-8, 10-4
HIPERCACHE 5-6
HIPEREDA 5-6
HIPEREXTENTS 5-6
HIPERFILE 5-6
HIPERINSTALL 5-6
HIPERLOCKED 5-6
HIPERSPACE 5-6
MVSLIM 5-8, 10-4
SERVLIM 5-8, 10-4
TCBLIM 5-8, 10-4

setting limits for IBI Subsystem 10-4

simultaneous usage feature 7-2
Resource Analyzer 7-12
Resource Governor 7-12

SMARTMODE parameter 7-8

SMCONTROL database 7-14

sorting tables for MVS 5-21

statistics 11-32 to 11-33
configuration file specifications 11-33
On Demand 11-33
region type 11-35
storage type 11-35

subserver connections 8-31

subserver tasks 11-38

SUBSYSI program 10-2
operator commands 10-3

subtask priorities 11-27
changing 11-27

SUPRA multi-session keywords 3-59

System Authorization Facility (SAF) 8-4 to 8-5

T

TASK parameter 3-59

TCBLIM parameter 5-8, 10-4

TCP/IP network definitions 2-15 to 2-16

TCP/IP parameters for servers 2-8
inbound 2-12, 8-26, 8-29
outbound 2-9, 8-33, 8-35 to 8-36

TCP/IP protocol blocks 8-23

TCP/IP requirements for MVS servers 2-3

TCP/IP service blocks 8-16, 8-18

THREAD parameter 3-59

TOP SECRET security system 8-7, 8-14

TRACE command 11-37

trace files for Resource Analyzer/Governor 7-16

Transaction Server for IMS 5-28 to 5-29
configuring 5-30 to 5-32
requirements 5-30
security 5-31

transcoding 5-16

Transcoding Services Generation Utility (TSGU) 5-16

troubleshooting IBI Subsystem 10-6 to 10-7

TSCFOLD procedure 6-4

TSGU (Transcoding Services Generation Utility) 5-16

TSO command line 10-3

U

URT (User Requirement Table) 3-4 to 3-5

usage accounting for Server for MVS 8-8

usage statistics for hiperspace 10-5

User Requirement Table (URT) 3-4 to 3-5

USERS command 11-38 to 11-39

user-written exits for CA-IDMS/DB 3-8 to 3-9, 3-12

V

viewing subserver tasks 11-38

VTAM APPLIDs 2-15

W

WHOHAS mode 11-11, 11-21

worksheets for ETL Server configuration 2-11

worksheets for Full-Function Server configuration 2-10 to 2-14

worksheets for Hub Server configuration 2-6 to 2-10

worksheets for server configuration 2-6

X

XMI servers 3-23 to 3-24, 3-26, 3-30
- CICS 3-29
- DLI mode 3-27
- FOCBMP communications files 3-25, 3-31
- PCB requirements 3-32
- stopping 3-32 to 3-33

XMISTOP job 3-32 to 3-33

Z

ZBIND exit 3-8 to 3-9

ZREADY exit 3-8, 3-12 to 3-13

Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail: Documentation Services - Customer Support
Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

Fax: (212) 967-0460

E-mail: books_info@ibi.com

Web form: http://www.informationbuilders.com/bookstore/derf.html

Name:_____

Company:_____

Address:_____

Telephone:_____Date:_____

E-mail:_____

Comments:

Reader Comments