

BrightStor™ CA-Vantage™ Storage Resource Manager ODBC/JDBC Interface

Installation and Operations Guide

62



Computer Associates™

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2001 Computer Associates International, Inc.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Introduction

Audience.....	1-2
The EDBC Solution.....	1-2
A Gateway Installation.....	1-3
The VANTAGE Gateway.....	1-4
Net.....	1-5
Ingres/Star.....	1-6
Security System Support.....	1-6
User Interfaces.....	1-7
Conventions.....	1-7

Chapter 2: Gateway Architecture and Operation

Gateway Subsystems.....	2-2
EDBC Address Space.....	2-2
Gateway Program Characteristics.....	2-2
Components of the EDBC Server Address Space.....	2-2
One or More Protocol Servers.....	2-3
A Single Name Server.....	2-4
A Multi-threaded Communications Server.....	2-4
One or More Gateways.....	2-4
Servicing Connection Requests.....	2-4
Accessing the Vantage Gateway.....	2-5
Achieving Access to Vantage Data.....	2-5
Registering Vantage Data.....	2-5
Storing Registration Information.....	2-5
Defining Relational Table Templates.....	2-6
Gateway Partitions and Areas.....	2-6
Sample Gateway Configuration.....	2-7
Summary: Satisfying Requests for Vantage Data.....	2-7

Chapter 3: Preparing for Installation

Software Requirements	3-1
Storage Requirements	3-2
Disk Storage	3-3
Virtual Storage Requirements	3-3
Installer's Requirements	3-4
Installer's Access to Resources	3-5
Installation Overview	3-6
Installation Summary	3-7
Installing EDBC and the Gateway	3-7
Verifying Gateway Functionality	3-7
Supporting Users	3-8
Registering Vantage Objects for Gateway Access	3-8

Chapter 4: Installing the Gateway

Installation Summary	4-1
Installation Framework	4-3
Installation Expectations	4-3
Product Tape	4-3
Backing Up Previous Installation Data Sets	4-4
Allocating and Loading the Gateway Data Sets	4-4
Creating JCL to Restore the Stage0 Jobstream	4-5
Customizing the Gateway Stage0 Installation Jobstream	4-5
Gateway Data Sets That Are Allocated and Restored	4-7
Customizing the Stage1 Input	4-8
IGWFJOB Statement: Specifying Jobcard Parameters	4-9
IGWFUSER Statement: Specifying Initial Gateway Users	4-10
IGWFNET Statement: Specifying Net Parameters	4-11
IGWFVANT Statement: Specifying Gateway Parameters	4-13
IGWFPSVR Statement: Specifying Protocol Server Parameters	4-15
IGWFBLD Statement: Specifying IIVP Sysgen Parameters	4-20
The Stage1 Jobstream	4-22
Customizing and Executing the IGWFSTGT JCL	4-22
Submitting the Stage2 Jobstream Jobs	4-23
Functions of Stage2 Jobs	4-24
Final Installation Procedures	4-32
Customizing the TSO Logon Procedure	4-32
Adding the Gateway Data Sets to the APF List	4-33

Performing an IPL of the OS/390 System.....	4-33
The Next Step.....	4-35

Chapter 5: Verifying the Gateway

Gateway Verification Summary	5-1
The Next Step.....	5-1

Chapter 6: Using the IGWFDBA Utilities

IGWFDBA Main Menu	6-1
Using This Menu	6-1
Option 0: Gateway Facility Profile Panel	6-3
Option 2: VSAM Gateway Facility Main Menu	6-6
Option 2.2: Display Active Gateway Drivers Panel.....	6-6
Option 2.4: Gateway Using File Input Panel	6-11
Option 2.5: Gateway Using Terminal Monitor Panel	6-12
Option 2.6: Create JCL to Invoke DI Utilities Panel.....	6-14
Option 2.6.1: Gateway Facility to Create JCL to Run diatedmp Utility Panel	6-15
Option 2.6.2: Gateway Facility to Create JCL to Run didirdmp Utility Panel.....	6-17
Option 2.6.3: Gateway Facility to Create JCL to Run dibackup Utility Panel.....	6-19
Option 2.6.4: Gateway Facility to Create JCL to Run direstor Utility Panel.....	6-21
Option 2.6.5: Gateway Facility to Create JCL to Run dicomprs Utility Panel.....	6-23
Option 2.6.6: Gateway Facility to Create JCL to Run dicopyar Utility Panel.....	6-25
Option 2.7: Create JCL to Format a Gateway Area Panel	6-27
Option 2.8: Create JCL to Create a User Partition Panel	6-28
Option 4: IIPARM Command Procedure Panel	6-30
Option 5: Net Utilities Menu Panel.....	6-31
Option 6: DBMS Utility Menu	6-33
Option 7: Create JCL to Compile ESQL Application Panel	6-35
Option 7.1: Create JCL to Compile Application in Whitesmiths C Panel.....	6-36
Option 7.2: Create JCL to Compile Application in PL/I Panel	6-39

Chapter 7: Configuring the EDBC Server

Install and Configure Communication Interfaces.....	7-1
SNA LU0 for OS/390	7-1
Requirements	7-2
Installation and Configuration	7-2
Starting and Stopping the SNA_LU0 Interface	7-2

Connecting from a Remote Client.....	7-3
SNA LU0 Abend Codes and Messages.....	7-3
SNA LU62 for OS/390.....	7-3
Requirements.....	7-4
Installation and Configuration.....	7-4
Starting and Stopping the SNA_LU62 Interface.....	7-4
Connecting from a Remote Client.....	7-5
SNA LU6.2 VTAM Logmode Entries.....	7-5
Sense Code 08120007 and Possible Loop in VTAM.....	7-6
KNET TCP/IP for OS/390.....	7-6
Requirements.....	7-6
Installation and Configuration.....	7-6
Starting and Stopping the TCP_KNET Interface.....	7-7
Connecting from a Remote Client.....	7-7
IBM TCP/IP for OS/390.....	7-7
Requirements.....	7-7
Installation and Configuration.....	7-8
Starting and Stopping the TCP_IBM Interface.....	7-8
Connecting from a Remote Client.....	7-8
IBM TCP/IP Problem Diagnosis.....	7-8
CCI for OS/390.....	7-9
Requirements.....	7-9
Installation and Configuration.....	7-9
Starting and Stopping the CCI Interface.....	7-9
Connecting from a CCI Client.....	7-10
CCI Abend Codes and Messages.....	7-10
SNS/TCP for OS/390.....	7-11
Requirements.....	7-11
Installation and Configuration.....	7-11
Starting and Stopping the TCP_SNS Interface.....	7-11
Connecting from a Remote Client.....	7-12
Enable and Test Security Interfaces.....	7-12
IBM Resource Access Control Facility (RACF).....	7-12
Installing and Customizing the RACF Interface.....	7-13
Computer Associates Access Control Facility 2.....	7-14
Installing and Customizing the CA-ACF2 Interface.....	7-14
Computer Associates Top Secret Security Facility.....	7-15
Installing and Customizing the CA-TSS Interface.....	7-15
Force Inactivate Timeout.....	7-17

Local Time Zone	7-17
Year 2000 Support	7-18
Alternate Translation Tables	7-18

Chapter 8: Setting Up Vantage Gateway Users

Gateway User Setup Summary	8-1
Expanding the Vantage User Base	8-2
Defining Needs of Gateway Users	8-2
Registering Vantage Data	8-2
Concepts: Area, Sort Data Set, Partition, and Driver	8-3
Area and Sort Work Data Sets	8-3
Partition	8-3
Using the IGWFDDBA Utilities	8-4
Adapting the Default Profile for a New User	8-4
Defining the Logical Symbols for the Gateway User	8-5
Formatting a Gateway User's Area	8-6
Gateway Partitions	8-8
Gateway Catalogs and the Gateway Partition	8-8
Creating a Partition	8-8
Checking Access to the Partition	8-10

Chapter 9: Registering Vantage Data

Vantage Objects and Relational Tables	9-1
Registration	9-2
Registering Vantage Objects	9-2
SQL Register Table Statement	9-2
Descriptions of Register Table Parameters	9-5
Relational Table Names	9-5
Column Descriptions	9-6
Data Type Mapping	9-6
EDBC Data Types	9-9
EDBC Character Strings	9-10
EDBC Numeric Data Types	9-12
Date Data Type	9-16
Money Data Type	9-20
Source of the Gateway Object	9-20
Access Methods and EDBC Structures	9-21
Rows	9-22
Repeating Groups	9-22

Repeating Group Restrictions.....	9-23
Registering a Table with a Repeating Group	9-23
The Occurs Data Type	9-24
Retrieving Data from a Repeating Group	9-24
Special Considerations When Working with Repeating Groups.....	9-24
VALUE Clause.....	9-25
Registering a Table with a VALUE Clause.....	9-25
Mapping Redefined Storage	9-26
Grants of Permission.....	9-27
Individual User Authorizations	9-28
Sharing Vantage Objects Among EDBC Users.....	9-28
Building Register Table Scripts	9-29
Testing Access to a Registered Table.....	9-30
Troubleshooting Registration Problems.....	9-31
Removing a Gateway Table or Index	9-32

Chapter 10: Maintaining the Gateway

Starting and Stopping the EDBC Server	10-1
Starting the EDBC Server.....	10-1
Stopping the EDBC Server.....	10-2
Bringing Up the EDBC Server as a Batch Job.....	10-3
Maintaining the EDBC Server Address Space	10-3
The Display Active Command	10-6
Logging and Recovery.....	10-8
Maintaining Areas.....	10-8
Showing Space Allocation in an Area	10-9
Showing File, Partition, and Area Directory Information	10-10
DI-File Analysis Scripts	10-11
Backing Up an Area.....	10-13
Restoring an Area.....	10-14
Increasing the Size of an Area	10-16
Compressing an Area	10-16
Copying Source Area to Target Area	10-17
Using the Gateway Utilities	10-19
createpr.....	10-19
Syntax	10-19
Description.....	10-19
Example.....	10-20

dbconst.....	10-20
Syntax	10-21
Description.....	10-21
destroyp.....	10-21
Syntax	10-21
Description.....	10-21
finddb.....	10-22
Syntax	10-22
Description.....	10-22
igwfczap	10-23
Description.....	10-23
verifydb	10-25
Syntax	10-25
Description.....	10-25
Example.....	10-27
vwconst.....	10-27
Syntax	10-27
Description.....	10-27
Defining an Additional EDBC Server.....	10-28
Using IIVP to Create a Second EDBC Server.....	10-28
Connecting from One EDBC Server to Another.....	10-30
Verifying the Connection	10-32
The Gateway Query Optimizer	10-33
The Query Execution Plan (QEP).....	10-34
Listing a QEP	10-34
Reading a QEP.....	10-35
Sample Tables for QEP Examples.....	10-36
Types of Nodes in a QEP.....	10-36
Orig Node.....	10-38
Proj-rest Node.....	10-38
Sort Nodes.....	10-39
Join Nodes	10-40
Multiple Query Plans.....	10-46
More Complex QEPS.....	10-48
Evaluating QEPS: A Summary	10-48
Optimizer Timeout.....	10-48
Debugging the Gateway	10-49
Tracing the Gateway Routing Exits	10-50
Tracing the Gateway Components.....	10-50
Diagnosing IGWF Initialization Failures After an IPL.....	10-50

Appendix A: Gateway Logical Symbols and IIPARM Clist

Logical Symbol Library Organization	A-1
Logical Name Format	A-2
Detailed Descriptions of Logical Symbols	A-2
ISVREDBC Logical Symbol Members	A-3
DBNMEDBC Logical Symbol Member	A-12
SABExxxx or Unnnnnnn Logical Symbol Member	A-13
Optional Logical Symbols	A-13
IIPARM Clist Description	A-14
Description of IIPARM Parameters	A-14
IIPARM Customization	A-16
Examples of IIPARM Use	A-16

Appendix B: Customization

EDBC.V2R3.SAMPLE.CNTL	B-1
EDBC.V2R3.FILES.PROCLIB	B-3

Appendix C: Accessing Vantage Data

Defining EDBC Server to EDBC Client	C-1
An ODBC Example: EDBC Client for Windows	C-2
ODBC Data Sources in Microsoft Windows	C-5

Appendix D: Example: Microsoft Excel

Accessing an ODBC Data Source	D-1
Selecting Tables	D-2
Define Filter Criteria	D-3
Define Sort Criteria	D-4
Displaying Vantage Data	D-5

Appendix E: Example: Web with FrontPage

Changing Document Type	B-1
Creating Web Page	B-3

Appendix F: Installing Multiple Gateways

Index

Introduction

BrightStor CA-Vantage Storage Resource Manager ODBC/JDBC Interface is part of a family of products that provides access to data across a wide range of hardware platforms, operating systems, and database management systems.

BrightStor CA-Vantage Storage Resource Manager (SRM) ODBC/JDBC Interface allows users to access data maintained by Vantage for OS/390. It also allows users on a client to develop database applications on a UNIX or Microsoft Windows platform that can be executed against Vantage data without requiring additional programming.

In this guide, the term:

- CA-Vantage gateway or gateway is synonymous with BrightStor CA-Vantage Storage Resource Manager ODBC/JDBC Interface
- OpenSQL is synonymous with EDBC OpenSQL
- Star is synonymous with Ingres/Star

The *BrightStor CA-Vantage Storage Resource Manager ODBC/JDBC Interface Installation and Operations Guide* serves as both a learning tool and a permanent reference guide.

This guide explains how to perform the following tasks:

- Prepare for installing the Vantage gateway
- Install the Vantage gateway on an IBM OS/390 system
- Verify that the Vantage gateway has been successfully installed
- Use the EDBC Gateway Facility Database Administrator (IGWFDDBA) utilities
- Operate and maintain the Vantage gateway on OS/390
- Work with the Vantage gateway from the perspective of an end user on a remote system

This guide provides an overview of the Vantage gateway and the other components with which the gateway interacts. It also covers the following topics:

- Gateway architecture and operation
- Configuring and starting EDBC

- Setting up Vantage drivers gateway
- Setting up Vantage gateway users
- Registering Vantage data using the Terminal Monitor

The appendixes include information on the following topics:

- Gateway logical symbols and IIPARM Clist
- Customization

Audience

The first six chapters of this guide are addressed to the individual who is responsible for installing the Vantage gateway and for supporting EDBC users who will access Vantage data through the gateway. It assumes a working knowledge of OS/390, TSO/E, ISPF, and JCL, plus a familiarity with Vantage for OS/390.

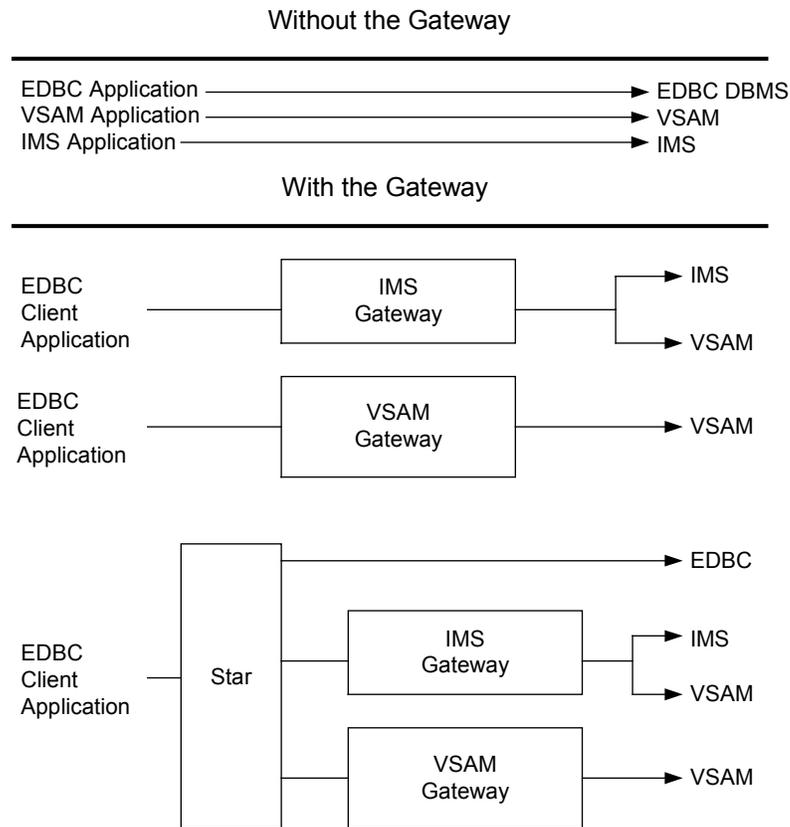
The “Introduction” chapter and the “Gateway Architecture and Operation,” “Configuring the EDBC Server,” and “Setting Up Vantage Drivers” chapters are directed to the end user who will be using remote applications to access Vantage data through the gateway.

This guide may also be useful to individuals who are already familiar with Vantage and OS/390 and who want to understand how the Vantage gateway works in this environment.

The EDBC Solution

EDBC for Vantage is one of a family of gateway products that provides a consistent way to access heterogeneous databases (logical or physical). The gateways permit EDBC users to run applications against local data and against data that resides in foreign relational and non-relational database management systems. *Relational gateways* provide bridges to relational database management systems such as CA-IDMS, CA-Datcom, Ingres, Rdb, IBM DB2, and ALLBase SQL. *Non-relational gateways* provide bridges to non-relational database management systems such as VSAM and IMS, and now to the database-like tables of information maintained by Vantage for OS/390.

The following figure shows an example of how a gateway provides these bridges:



A Gateway Installation

A gateway installation, running under OS/390 at a Vantage site, includes the following components:

- Gateway to access Vantage tables
- EDBC to permit access between OS/390 and user interfaces residing on a different computer system
- Star to allow an EDBC client to access multiple databases simultaneously
- Several client user interfaces, such as the Terminal Monitor, ODBC, ADO, and OLE/DB

All components communicate through the Global Communications Architecture (GCA), which resides at each client and server location.

The following sections describe these components in detail.

The Vantage Gateway

Vantage tables are repositories for a wealth of information about your OS/390 environment. The gateway gives EDBC clients transparent access to Vantage data. It also enables programmers to use local client tools to develop and deploy applications that process Vantage data.

Combined with other components, the gateway:

- Provides transparent access to Vantage data
The gateway allows you to run client user interfaces to access tables and views stored in a Vantage table. This access is transparent to the end user. The Vantage data appears to an EDBC client as though it is stored in a local database.
- Exploits the standard Vantage EWS Message Layer
The gateway uses the standard Vantage EWS Message Layer, which is also the facility used by the Vantage GUI, translating queries from EDBC client user interfaces into native Vantage EWS requests. Vantage executes the queries and returns the results back to the gateway. The gateway collects and packages the responses into syntax meaningful to the client user interfaces.
- Enhances programmer productivity
The gateway allows a client to build portable database applications for Vantage using client-based application development tools, such as Power Builder, FrontPage or Visual Basic. These applications can run unmodified against any server or gateway, and can coexist with existing Vantage applications that access and update this data.
- Unifies data access with a single language
The gateways allow users to work with data using ADO. This allows them to access Vantage, VSAM, Ingres, or any other DBMS using the same language.
- Integrates desktops, workstations, and foreign mainframes into the OS/390 environment
The gateway allows EDBC clients to retrieve and process foreign data, and develop database applications while working on computers such as Sun, HP, and PCs running under operating systems such as UNIX and Microsoft Windows.
- Supports distributed processing
With the addition of Star, the gateway allows users to join data from diverse databases, such as CA-IDMS, CA-Datcom, Ingres, IMS/VS, VSAM, Vantage, and IBM DB2, and use the resulting tables as if they were derived from a single source.

Note: Star is not required to join IMS and VSAM data.

Net

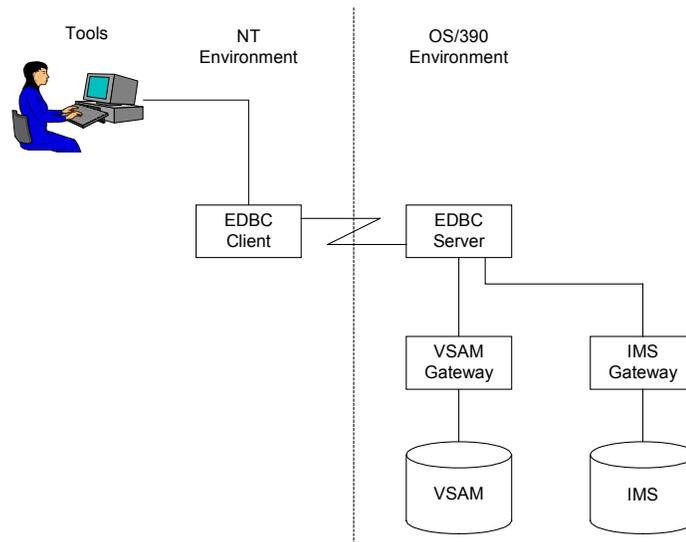
EDBC allows you to access data in two modes:

- In *local* mode, the EDBC user interfaces and the target database reside on the same computer system.
- In *remote* mode, the EDBC user interfaces and the target database reside on different computer systems. These systems must be linked with a computer network. The system where the user interfaces reside is called the *client*. The system where the target database, and, if necessary, the gateway, resides is called the *server*.

Net is a network application that enables EDBC users to access databases on remote platforms. With the gateways, Net permits communication between components running on many types of machines, under diverse operating systems, including OS/390, UNIX, and Microsoft Windows.

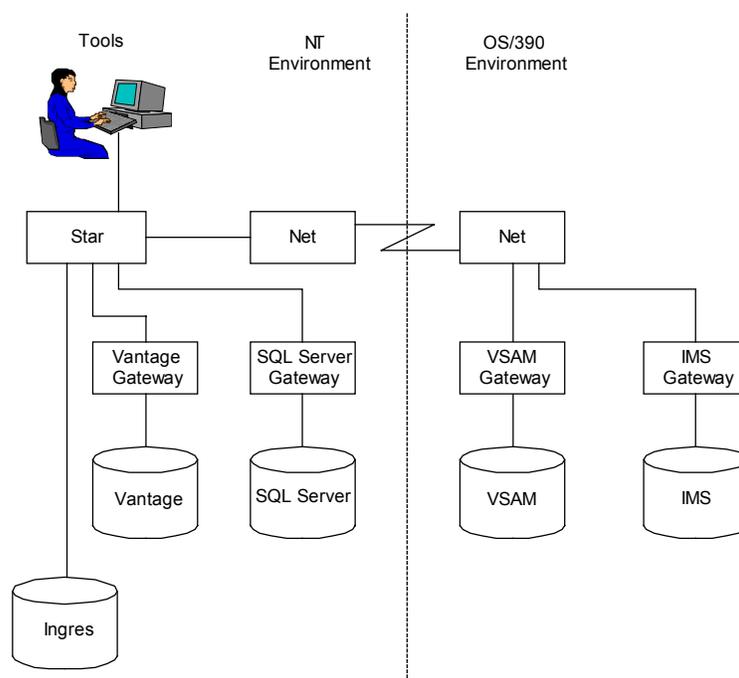
The Net software is installed on each machine that represents a node on the network. When the user invokes an EDBC tool or application on his local node, Net passes the query to the database on the proper remote node, using the appropriate protocol. Net supports multiple protocols, including SNA LU0, SNA LU62, TCP/IP, DECnet, PC LANs, NetBIOS, Interlink SNS/TCP and CCI. CCI is supported only on OS/390.

The following illustration shows a Net and gateway configuration connecting an EDBC installation on an NT system with an OS/390 system:



Ingres/Star

Star is a distributed information manager that allows an EDBC user to access multiple databases simultaneously. Star makes it possible to create a distributed database containing data from EDBC databases and repositories such as CA-IDMS, CA-Datcom, IMS, RMS, VSAM, Vantage, and IBM DB2. From a user's perspective, a distributed database (DBD) has all the characteristics of a local relational database. The base tables, however, remain physically located in multiple databases that can be of one or more types, residing on one or more systems. The following illustration shows a configuration similar to the one in the previous illustration, but with the addition of Star. See the *Star User Guide* for additional information about Star and distributed databases.



Security System Support

The gateway fully supports the following major OS/390 security systems:

- IBM Resource Access Control Facility (RACF)
- Computer Associates Access Control Facility 2 (CA-ACF2)
- Computer Associates Top Secret Security (CA-TSS)

For more information about these systems, see the "Configuring the EDBC Server" chapter.

User Interfaces

Any application development interface that is ADO, OLE/DB, ODBC or JDBC compliant can be used to access Vantage data through the gateway.

The gateway supports access from EDBC user interfaces to Vantage tables. However, the converse is not true. A user running a Vantage application cannot use EDBC to access data stored in another database.

Conventions

This guide employs several conventions, described in this section, to make identifying information easier.

Terminology

This guide observes the following distinction in terminology:

- A *command* is an operation that you execute at the operating system level. An extended operation invoked by a command is often referred to as a *utility*.
- A *statement* is an operation that you embed within a program or database procedure, or execute interactively (for example, using the Terminal Monitor or the SQL Test window in Visual DBA).

A statement can be written in Ingres/4GL, a host programming language (such as C), or a database query language (such as SQL or QUEL).

Query Languages

The industry standard query language, SQL, is used as the standard query language throughout this guide. The gateway is compatible with applications compliant with ISO Entry SQL92.

EDBC operates on data using the Structured Query Language (SQL). SQL consists of high-level descriptions of actions to be performed against data, such as select and prepare. SQL requires no instructions for accessing data; it navigates its own way through the database.

The gateways use OpenSQL, which is a subset of Ingres/SQL. OpenSQL is highly portable, because it is designed to open applications to many different types of databases. All EDBC products support OpenSQL. Applications written in SQL must be rewritten in OpenSQL for use with the gateways. OpenSQL is documented in the *EDBC OpenSQL Reference Guide*.

Syntax and User Input When representing syntax and user input, the following conventions are used:

Convention	Usage
Boldface	Indicates any text that you must type as shown.
<i>Italics</i>	Indicates a variable name or placeholder for which you must supply an actual value – this convention is used in explanatory text, as well as syntax.
Case Sensitivity	System command and environment variable names may be case-sensitive, depending on the requirements of your operating system.
[] (square brackets)	Used to enclose an optional item.
{ } (curly braces)	Used to enclose an optional item that you can repeat as many times as appropriate.
(vertical bar)	Used between items in a list to indicate that you should choose one of the items.

Example The following example illustrates some of these conventions:

cd *directory*

The **cd** portion is in bold, so you enter it exactly as shown. The *directory* portion is in italic, indicating a placeholder that you must replace with a value that makes sense in your own system.

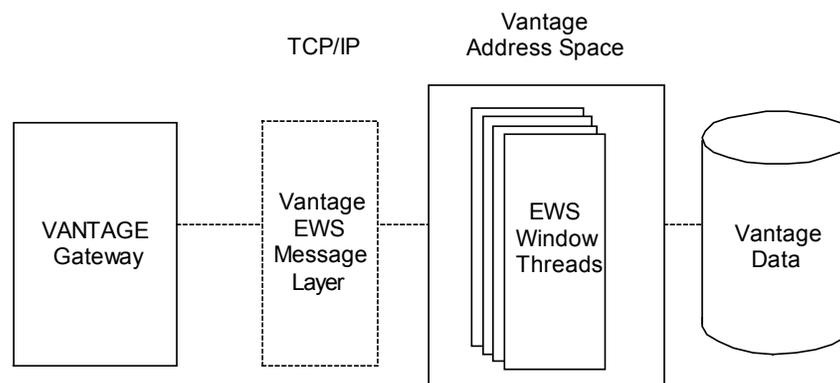
Gateway Architecture and Operation

This chapter provides a technical overview of EDBC for Vantage. It is useful to read this overview before beginning the installation because the chapter explains terminology and concepts that are used throughout the guide.

The chapter:

- Introduces the components of the Vantage gateway
- Summarizes the process of registering Vantage data as tables and indexes, and explains how the resulting information is stored in gateway areas and partitions
- Describes how components interact with Vantage to service an EDBC user's request for Vantage data

The Vantage gateway consists of several major entities, as shown in the following figure:



Gateway Subsystems

The Vantage gateway is installed as a single subsystem in the form of the EDBC server. This gateway subsystem has an associated address space, which may include other EDBC gateways (such as the VSAM, IMS, or DB2 gateways to name only a few) or may be configured to only support the Vantage gateway.

EDBC Address Space

The gateway components reside in a common address space, the EDBC server address space. The Vantage data may be accessed directly from the EDBC server address space; this is referred to as *local* access and requires no other address spaces.

The entire EDBC server address space is brought up and shut down as a unit.

Gateway Program Characteristics

The gateway software is implemented as multiple threads that are dispatched separately. On a single CPU, the threads run concurrently. In a multi-processor system, these threads can be dispatched to different CPUs to enhance throughput. The executable load modules are re-entrant, so multiple threads share a single copy of the executable load module.

The EDBC server subsystem (and its associated address space) runs as non-swappable code. This reduces page faults during operation. The executable load modules required for the gateway can be loaded in the Link Pack Area (LPA) that MVS reserves for frequently used programs.

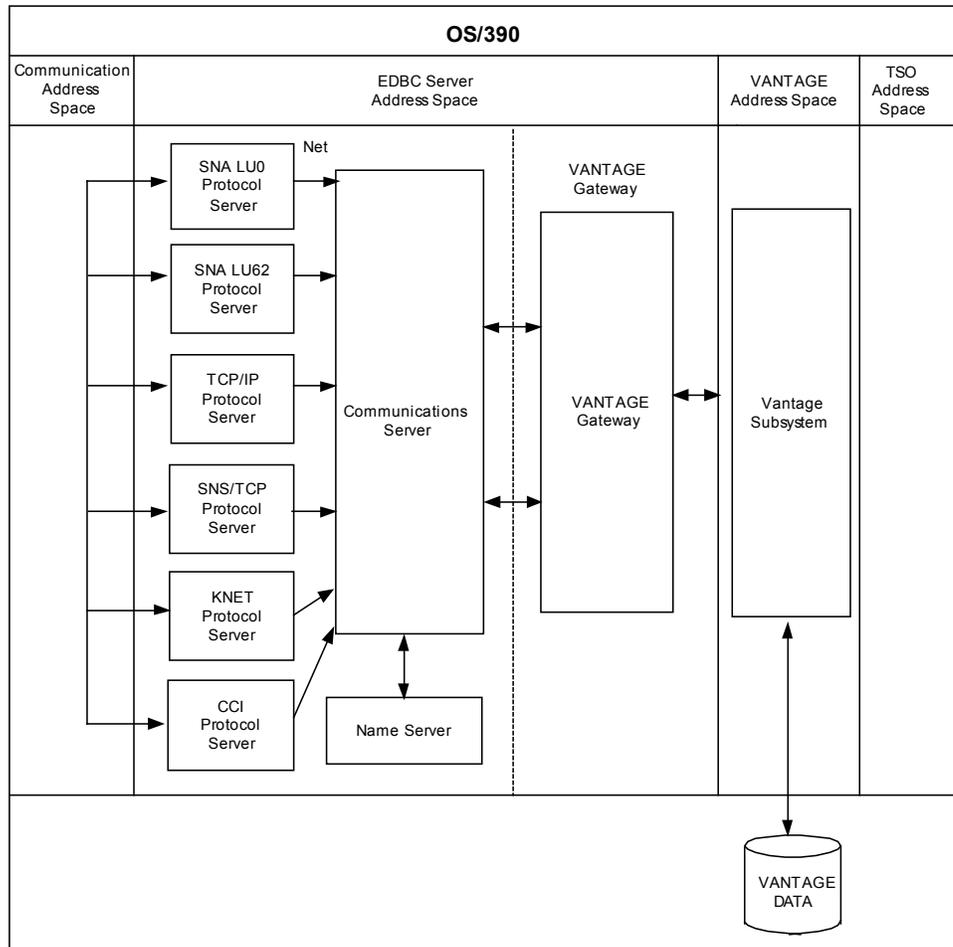
The gateway code runs above the 16 MB line. The EDBC server address space requires 640 bytes of Common System Area (CSA) storage below the 16 MB line for intra-address communication.

EDBC clients establish network connections to the OS/390 host through communications software (VTAM or TCP/IP). The Vantage gateway code requires TCP/IP to connect to active Vantage subsystems, even if a target Vantage subsystem resides on the same OS/390 image as EDBC server.

Components of the EDBC Server Address Space

The Vantage gateway executes in a high performance, multi-threaded server. You can create additional EDBC servers, if necessary, to increase the total number of concurrent users, to isolate groups of users for site-specific reasons, or to provide load balancing.

The following figure shows the major components of the gateway and the way they interact with the Vantage drivers and Vantage:



The EDBC server address space contains the components described in the following sections.

One or More Protocol Servers

Six protocol servers monitor traffic over the network. These servers recognize and process incoming communication requests in their native protocols, SNA LU0, SNA LU6.2, IBM TCP/IP, SNS TCP/IP, CCI, and KNET TCP/IP. The protocol servers provide a multi-threaded interface to the underlying physical transport. They communicate with EDBC through in-memory message queues.

A Single Name Server

A single name server queues incoming connection requests and handles them serially. The name server maintains a list (of the Vantage, and perhaps, the VSAM, IMS and/or DB2 gateways) that is installed in the EDBC server address space.

A Multi-threaded Communications Server

A multi-threaded communications server (Comm server) establishes connections to the name server and the EDBC server. A new Comm thread is created for each active connection, whether incoming or outgoing. In addition, one Comm server thread listens for new communications requests. The Comm server provides communication protocol stack functions and transport functions, performs data format conversion, and manages session context. The Comm server thread services a connection to a gateway thread.

One or More Gateways

The following gateways can co-exist within the same EDBC server address space:

- A Vantage gateway
- A VSAM gateway
- An IMS gateway (if you are running EDBC for IMS)
- A DB2 gateway (if you are running EDBC for DB2)
- A CA-IDMS gateway (if you are running EDBC for CA-IDMS)
- A CA-Datacom/DB gateway (if you are running EDBC for CA-Datacom/DB)

Servicing Connection Requests

For each new connection request, the EDBC server address space creates a new Comm server thread and a new gateway thread, and thus defines a new gateway environment. So, for each new connection, Vantage sees a separate and distinct application program.

After a new Comm server thread and gateway thread are started up, these remain active for the duration of the connection.

Accessing the Vantage Gateway

The Vantage gateway can be accessed in two different manners: it can be accessed via the EDBC server (server mode) or it can be accessed independently of the EDBC server (via stand-alone mode). Only clients running on the MVS mainframe can use stand-alone mode. When running the gateway in stand-alone mode, the EDBC server is bypassed and the Vantage gateway is invoked directly in the MVS user's TSO or batch address space. The IIPARM logical ING_MODE determines the mode in which the gateway is to be accessed.

Achieving Access to Vantage Data

This section describes the process of registering Vantage data and achieving access to it through the Vantage gateway.

Registering Vantage Data

In Vantage, data is in logical database tables known to Vantage as "objects"; in SQL-based applications, data is relational. To enable EDBC users to access Vantage data, the Vantage objects must be registered.

To register Vantage objects as EDBC tables, the Vantage object formats must be mapped to EDBC table definitions.

This should be done through the use of the VANTEDBC scripting utility. This scripting utility is used to connect to existing Vantage subsystems to query these subsystems, gather information about the internal Vantage objects, and to generate EDBC scripts that may be used to define these objects as EDBC tables.

Storing Registration Information

The data registration process produces a series of table templates, one or more for each registered Vantage object. These table templates are stored in the gateway partition for the EDBC user who is given access to these files. The gateway partition is a storage area created to hold information about EDBC relationships and entities.

The table templates are stored permanently in the gateway partition, but the actual data resides only in the Vantage object. The EDBC server accesses the physical Vantage data but does not copy it into the partition. Instead, each time it receives a request for data, the gateway extracts the data from Vantage.

Defining Relational Table Templates

Vantage objects have the following relationships to each other:

- Each Vantage object is registered as a separate relational table.
- Each record in a Vantage object corresponds to a row in the relational table.
- Each field in a record corresponds to a column in the table.

One major aspect of the registration process is defining the layout of the desired relational table—that is, specifying the column headings and data types that the user expects to see. Then, when a user queries this segment, the Vantage data is mapped into the prescribed table format. Where necessary, data types are converted from the Vantage to the specified data type.

Each object may be registered one or more times, depending upon which field of the object is to be used as the “key” field. A registered object can be removed, if it is no longer required. You need to remove and reregister a object only when its structure changes, not when its contents change.

Gateway Partitions and Areas

The Vantage gateway installation process creates a data dictionary. The data dictionary consists of a set of tables called the *standard system catalogs*. These catalogs are stored in the gateway area. The data registration process populates these system catalogs with information about the Vantage objects.

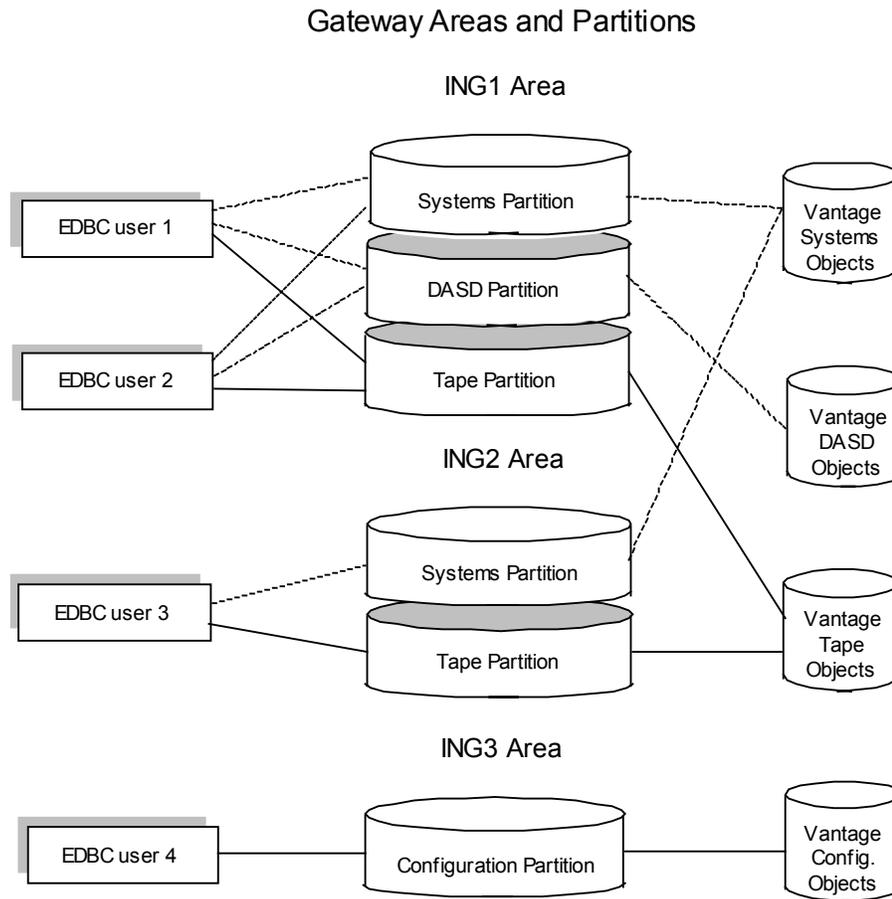
Partitions are allocated dynamically from within a formatted gateway area. Associated with each area is a sort work file, which is used to support relational processing

The area and the sort work file are both VSAM linear data sets (LDS) with a control interval size of 4096 bytes and a block size of 4096 bytes. Linear data sets are not device dependent and can reside on any cataloged device. The gateway uses the area LDS to store temporary tables and maintain system catalogs on disk. You can back up and restore these files with standard IBM utilities.

Multiple users can share the same area.

Sample Gateway Configuration

The following figure shows a possible configuration of gateway areas and partitions for an installation with three EDBC users:



For example, one EDBC user is given the area named ING1. This area contains three partitions, because the user works with three applications that utilize different sets of Vantage data. Another EDBC user has an area named ING2, with partitions that reference two of the three groups of objects used by ING1. A third user ID has area ING3. This contains a single partition, referencing a group of files the other users cannot access.

Summary: Satisfying Requests for Vantage Data

When an EDBC user queries a given Vantage object, the EDBC server checks to see whether the Vantage object has been defined to the partition.

If so, EDBC server invokes the Vantage interface code with specific information about the user's query. The Vantage interface code turns this information into standardized EWS Message Layer requests, which are sent over to an active Vantage subsystem via TCP/IP. The Vantage interface processes the EWS responses and feeds the requested information back into EDBC server to be processed against the users query and ultimately returned to the EDBC user.

Preparing for Installation

Before you install the Vantage gateway, you need to prepare for the installation. This chapter describes these preparations as follows:

- Outline the hardware, software, and storage requirements for installing the gateway components on OS/390
- Outline the privileges and knowledge required of the gateway database administrator
- Address the key aspects of configuring the networking software
- Summarize the major tasks that are required to install the gateway components on OS/390 that enable EDBC users to access Vantage objects

Read through this chapter to get an overview of all phases of the installation. In addition, be sure to read the *Release Notes* for your current release. The *Release Notes* update the information in this guide.

Software Requirements

The Vantage gateway requires the software resources listed in the following table:

Software	Specifications
IBM Operating System	Any currently supported release of MVS/ESA, OS/390, and z/OS
Application Subsystem	TSO/E
Communication Access Method: VTAM	ACF/VTAM
One of the following communication protocols:	
■ SNA LU0	
■ SNA LU62	

Software	Specifications
■ KNET TCP/IP	
■ IBM TCP/IP	
■ CCI	
■ SNS/TCP	
■ IBM TCP/IP for EWS	Initial release requirement
DB/DC System: CICS (if using CICS driver)	
Security Facilities	<p>The Vantage gateway supports any of the following security facilities:</p> <ul style="list-style-type: none"> ■ IBM Resource Access Control Facility (RACF) ■ Computer Associates Access Control Facility 2 (CA-ACF2) with support for IBM's Security Access Facility (SAF) ■ Computer Associates Top Secret Security (CA-TSS)
User interfaces on client platform	<p>Supported interfaces include:</p> <p>EDBC Client Ingres Client Jasmine Client</p> <p>The client can reside on Microsoft Windows or any of more than 30 UNIX platforms.</p>
Gateway user interfaces on OS/390 platform	<p>Netu SQL Terminal Monitor Embedded SQL (ESQL) Preprocessor for C ESQL for PL/I</p>

Storage Requirements

The gateway requires the storage specified in the following sections.

Disk Storage

The gateway data sets require 300 cylinders of space on a 3380 or 3390 storage device.

If an OS/390 archive product such as CA-ASM2 or IBM's DFHSM is installed, exclude the gateway data sets from migration.

Virtual Storage Requirements

EDBC Server Address Space	<p>The EDBC server and the Vantage gateway are installed together in a single OS/390 address space. Most of the code runs above the 16 MB line. The EDBC server address space requires 640 bytes of Common System Area (CSA) storage below the 16 MB line for intra-address communication.</p> <p>Supporting 250 concurrent connections requires a region size of approximately 8 MB. When a new connection is established, it acquires a control block and about 764 KB of virtual storage above the line. The storage is allocated for the life of the connection and returned when the user disconnects. An individual connection requires no additional storage in terms of Common System Area (CSA) storage below the 16 MB line.</p> <p>During initialization, the EDBC server address space requires a working set of about 5 MB. The initialization phase occurs only once, during startup, and takes about a minute. While the gateway is dormant, the working set is about 300 KB.</p> <p>The programs that run in the EDBC server address space are re-entrant, and the EDBC server address space itself runs as non-swappable code. If you wish, you can load this code in the Link Pack Area (LPA) that MVS reserves for frequently used programs.</p>
User Address Space	<p>As the gateway DBA, you can access the gateway from OS/390 using the EDBC Gateway Facility DBA (IGWFDDBA) utilities that are provided for gateway installation and support. When you access the gateway locally, the gateway and user interfaces, such as the Terminal Monitor, run in your user address space (batch or TSO). This requires a region size of 4MB to 8 MB.</p>
Gateway Areas and Partitions	<p>The installation process automatically formats a gateway area, a sort work linear data set, and a partition for the gateway DBA. You can subsequently create additional areas, sort work data sets, and partitions. A gateway utility allocates space dynamically within an area to build each partition that you define.</p>

For a 3380 DASD, the default size for a gateway area is 2700 records, as measured in 4 KB blocks. The default size for the sort work data set is 150 records, in 4 KB blocks. The desired values in the JCL that creates these linear data sets can be specified. If a user will be carrying out complex queries like four-way joins on large amounts of data, or if the user requires more than two or three partitions, a larger area and sort work data set may be required.

Gateway areas and sort work data sets are linear data sets created for the gateway. They can be saved and restored with standard IBM utilities. Specify an initial storage location for each area that will minimize fragmentation of the disk pack. If you need to, however, you can use the IBM utilities to back up and restore these data sets later to a different location.

Installer's Requirements

The Vantage gateway requires that someone on OS/390 installs the gateway initially and provides ongoing support for EDBC users who are working with the gateway. The person who assumes these responsibilities is referred to throughout this guide as the gateway database administrator (DBA).

Installing the Vantage gateway and EDBC server requires familiarity with VSAM as well as a working knowledge of OS/390, TSO/E, ISPF, and JCL.

To expedite installation, the gateway DBA should have:

- Global system administrator status within OS/390, with authority to:
 - Update the OS/390 system data sets listed in the Installer's Access to Resources section in the "Preparing for Installation" chapter.
 - Schedule or perform an IPL of OS/390.

If necessary, the installation process can perform the required customization and place the files in gateway target libraries. Later, someone with the necessary authority can copy these files to system libraries.

- Ability to configure the software protocol for EDBC, and either SNA LU0, SNA LU62, KNET TCP/IP, IBM TCP/IP, CCI, or SNS/TCP
- Ability to create OS/390 user IDs and authorize them for TSO and Vantage object access. These user IDs are assigned to the EDBC users who will access the gateway
- Authority to create the sample VSAM data sets that are part of the gateway installation and verification process
- Authority to access Vantage objects through normal Vantage security facilities (see the Vantage documentation for information on Vantage security measures)

Supporting EDBC users is an ongoing responsibility that is carried out with a series of gateway utilities called the EDBC Gateway Facility DBA (IGWFDBA) utilities. These menu-driven utilities are ISPF panels that automate the generation of JCL to perform administrative operations. Using these utilities does not require any knowledge of the gateway or SQL beyond the information provided in this guide.

Installer's Access to Resources

To install the gateway on OS/390, you must access or update a number of resources in OS/390 and Vantage. These standard system libraries and utilities are listed in the following tables. If you do not have the necessary authority, you can prepare the JCL and control statements and have the updates made by the appropriate individual.

The following table lists standard system libraries:

Libraries Referenced	Library Name	Access Mode
OS/390	SYS1.LINKLIB	Execute and optional update
	SYS1.MACLIB	Read
	SYS1.PROCLIB	Execute and optional update
	SYS1.SORTLIB	Execute
	SYS1.VTAMLIB	Optional update
	SYS1.VTAMLST	Optional update
	SYS1.SISTMAC1	
	SYS1.AMODGEN	
	SYS1.MODGEN	

The following table lists standard system utilities:

System	Utility Name	Access Mode
OS/390	IDCAMS	Execute
	IEBUPTTE	Execute
	IEFBR14	Execute
	IEV90	Execute
	IEWL	Execute
	IKJEFT01	Execute

Installation Overview

The gateway installation procedure is modeled on a standard OS/390 installation procedure. The installation process uses TSO to customize JCL and control statements that are on the gateway product tape. The estimate of the time required to install the gateway is approximately eight hours.

The installation procedure is referred to as the EDBC Installation and Verification Procedure (IIVP). The IIVP installs EDBC server, the Vantage gateway, and the gateway user interfaces on OS/390 (the SQL Terminal Monitor, the embedded SQL Preprocessor for C, and the embedded SQL Preprocessor for PL/I).

The installation process also creates sample VSAM data sets and maps their record layouts into registered tables. These data sets are provided so that the gateway DBA can verify that the gateway has been successfully installed. Operating locally under TSO, the gateway DBA issues SQL queries to the sample data sets. The resulting data is presented in relational tables, just as it would appear to an EDBC end user.

The gateway DBA performs the following tasks to set up the environment for each EDBC user:

- Creates and authorizes OS/390 user IDs to access Vantage objects and, optionally, TSO
- Formats additional gateway areas and sort data sets
- Creates additional partitions

Once EDBC has been installed, the DBA on the remote system tests access to the gateway across the network. Finally, gateway users on the remote system run netu to define any private accounts established on their behalf under OS/390.

Installation Summary

The gateway installation is done in the phases listed below. The chapter references indicate where the details of each phase are described in this guide:

- Installation of the EDBC server and the gateway
- Verification of the installation
- Setting up of Vantage drivers (“Setting Up Vantage Drivers” chapter)
- Setting up of Vantage gateway users (“Setting Up Vantage Gateway Users” chapter)
- Registration of Vantage objects (“Registering Vantage Data” chapter)

A summary of the steps in each phase is presented in the following sections.

Installing EDBC and the Gateway

This section summarizes the procedure for installing the EDBC and gateway components on OS/390. The steps in the installation phase are as follows:

1. Allocate and load the EDBC data sets from the product tape.
2. Customize the macros in the file that contains the stage1 input.
3. Customize and submit the JCL to take the stage1 input file and produce the stage2 jobstream.
4. Submit the stage2 jobstream for execution, releasing each job one at a time and checking the return codes.
5. Install the IGWF subsystem permanently.
6. Customize the OS/390 system by adding two OS/390 subsystems, one for the EDBC server and one for IGWF, then adding the EDBC data sets to the APF list.
7. Modify the TSO logon procedure for the gateway DBA.
8. IPL OS/390, if required.

See the “Installing the Gateway” chapter for a detailed description of the installation procedure.

Verifying Gateway Functionality

Once the gateway is installed, you can test the gateway by accessing the sample data sets that were created during the installation. Initially, verification will be performed using the VSAM gateway. Later, you will verify proper configuration of the Vantage gateway by accessing actual Vantage data.

To verify the gateway installation, you must invoke the Terminal Monitor from TSO and use it to test local access through the gateway to the sample data sets.

See the “Verifying the Gateway” chapter for more information about this procedure.

Supporting Users

For users to access the gateway, you need to complete the following steps:

1. Create the necessary OS/390 user IDs and authorize their access to appropriate Vantage objects and, optionally, TSO.
2. Using the IGWFDBA utilities, create a profile and a set of gateway logical symbols for each unique user.

The logical symbols define important start-up and run-time parameters, and configure the gateway environment properly for each user.

See the “Setting up Vantage Drivers,” “Setting up Vantage Gateway Users,” and “Registering Vantage Data” chapters for a detailed description of the procedures to complete these steps.

Registering Vantage Objects for Gateway Access

The following steps summarize how to register Vantage data:

1. Edit the sample VANTEDBC JCL to run against an active Vantage subsystem to create the necessary EDBC registration scripts.
2. VANTEDBC creates members in a PDS of your choosing, one for each Vantage object.
3. VANTEDBC also creates a member named \$RUNJOB. Edit this member and follow the instructions defined in JCL comments found at the front of this JCL.
4. Submit the customized \$RUNJOB jcl to automatically register every Vantage object to the EDBC gateway.
5. Test access to the registered tables locally, from TSO or the SQLDBA utility.

See the “Registering Vantage Data,” chapter for a detailed description of the registration process.

Installing the Gateway

This chapter provides step-by-step instructions for installing the Vantage gateway and EDBC server on OS/390 using the EDBC Installation and Verification Procedure (IIVP).

This installation phase is organized into three major stages:

- Customizing the stage1 input and producing the stage2 jobstream
- Submitting the stage2 jobstream jobs
- Completing the final installation procedures

Many of the tasks involve customizing JCL and control statements provided on the gateway product tape. Most of the work is performed under TSO.

Installation Summary

The following table shows a summary of the installation tasks and their output:

Task	Output
1. Back up previous Vantage gateway installation, if necessary.	Tape containing previous installation files and any user applications stored in front end catalogs.
2. Create and submit IEBGENER job to restore bootstrap JCL from tape.	Stage0 installation jobstream restored into data set with default name EDBC.V2R3.STAGE0.JOB
3. Edit and submit stage0 jobstream to allocate and copy gateway data sets on to disk.	Gateway installation data sets, with default names such as: EDBC.V2R3.FILES.ASM EDBC.V2R3.FILES.MACLIB EDBC.V2R3.FRONT.LOAD

Task	Output
<p>4. Use ISPF to customize statements in stage1 input files to reflect installation-dependent values:</p> <ul style="list-style-type: none"> ■ IGWFJOB ■ IGWFUSER ■ IGWVANT ■ IGWFDRIV ■ IGWFSGRP ■ IGWFDEMO ■ IGWFPSVR ■ IGWFBLD 	<p>Customized and saved statements in member IGWFSTGT in data set with default name EDBC.V2R3.FILES.ASM.</p>
<p>5. Customize and submit assembly JCL that uses stage1 input to produce stage2 jobstream. Input is from member IGWFSTGT in data set with default name EDBC.V2R3.SAMPLE.CNTL.</p>	<p>Stage2 jobstream, with jobs, report, and listing.</p>
<p>6. APF authorizes the following libraries (the names specified are the default names):</p> <p>EDBC.V2R3.BACK.LOAD EDBC.V2R3.NET.LOAD</p>	<p>The libraries can be APF authorized temporarily using the OS/390 SETPROG APF operator command. However, the libraries should eventually be added to the IEAAPFxx member of SYS1.PARMLIB to make them permanently APF authorized.</p>
<p>7. Submit jobs in IIVP stage2 jobstream. The jobstream is located in the data set with the default name EDBC.V2R3.STAGE2.CNTL.</p> <p>Release jobs one at a time from hold status in the following sequence:</p> <p>IGWVPA1 IGWVPA9 VANTA0 VANTIO VANTN0 VANTT0 VANTT1 VANTT2 VANTT3</p>	<p>Customized files that build the gateway environment, creating load module, authorizing cataloged procedures, etc.</p>
<p>8. Customize TSO logon procedure for gateway DBA.</p>	<p>Saved logon procedure.</p>

Installation Framework

The Vantage gateway requires one OS/390 subsystem, the EDBC server. The gateway subsystem has the default name EDBC. It is recommended that you accept this default name, as this facilitates installation.

You do not need to IPL the OS/390 system to test the Vantage gateway. Read through this entire chapter to get an overview of the process before you begin the installation. In addition, be sure to read the Readme file.

Installation Expectations

The IIVP stage2 jobstream accomplishes the following operations without an IPL:

- Customizes the installation and runtime parameters.
- Customizes the EDBC Gateway Facility DBA (IGWFDDBA) application ISPF dialog data sets.
- Builds and initializes a gateway area that is used to store gateway partitions.
- Dynamically identifies Vantage data objects to the Vantage gateway.
- Invokes the Terminal Monitor access to Vantage data through the Vantage gateway to verify the install.

The following operations are *not* performed by the IIVP stage2 jobstream:

- Adding JCL passwords to any jobs. Use an installation-approved method to provide this information.
- Adding the gateway data sets to the APF list in IEAAPFxx.
- Updating the TSO logon procedure to add the data sets required by the IGWFDDBA application. The gateway DBA must use this updated logon procedure.
- Installation of the EDBC client. This must be done before the gateway can be accessed from remote clients.

Product Tape

The product tape contains all the gateway files needed to install the Vantage gateway. Install this product separately from any previously existing gateway products. You can migrate this product to production status at a convenient time.

Note: You must use the macros distributed on this Vantage gateway tape for this product's installation and customization.

The Vantage gateway and Net are distributed on a single 9-track tape. The tape has the following characteristics:

- IBM 3240 type or IBM 3480
- Standard label format, made using the IBM utilities IEBCOPY and IEBGENER
- Contains 58 files
- VOLSER printed on the external label of the tape (required information for installation procedure)
- Contents require approximately 300 cylinders of storage on a 3380 or 3390 DASD

Backing Up Previous Installation Data Sets

If there is an existing gateway installation, either back up the existing gateway files or assign a different data set prefix to the new gateway files.

It is recommended that you back up the following files:

```
EDBC.V2R3.FILES.ASM  
EDBC.V2R3.SAMPLE.CNTL  
EDBC.V2R3.FILES.SQL  
EDBC.V2R3.FILES.IIPARM  
EDBC.V2R3.VANT.REGISTER*
```

*This file is new to EDBC V2R3 and may not be available currently.

To back up files for an existing gateway installation:

- Save any register table scripts you want to keep.
- Alert the DBA on the remote system to use an appropriate user interface to back up any front-end objects that are stored in existing gateway partitions on OS/390.

Caution! If you assign the new gateway files the same data set prefix as the existing files, the old files will be deleted before the new files are loaded.

Allocating and Loading the Gateway Data Sets

The first file on the gateway product tape file is a bootstrap jobstream that prepares for the allocation and copying of the installation data sets. After the stage0 jobstream has been restored from File 1 of the tape, customize and submit the jobstream for execution. The stage0 jobstream then allocates and loads all the remaining gateway data sets from tape to disk pack.

Creating JCL to Restore the Stage0 Jobstream

Create a job called IGWFJOB0 that uses the IEBGENER utility to allocate and copy File 1 from the gateway product tape to a disk data set. Use the following procedure to do so:

1. In an OS/390 data set, create JCL like the sample shown below:

```
//IGWFJOB0 JOB
//*
//* ALLOCATE AND COPY FILE 1 TO A DISK DATASET
//* THE RESTORED FILE IS THE STAGE 0 JOBSTREAM
//*
//COPYFILE      EXEC  PGM=IEBGENER
//SYSIN          DD   DUMMY
//SYSPRINT      DD   SYSOUT=*
//SYSUT1        DD   DSN=EDBC.INSTALL.CNTL,
//              VOL=(,RETAIN,SER=IVxxyy),
//              UNIT=TAPE,LABEL=(1,SL),
//              DISP=(OLD,KEEP)
//SYSUT2        DD   DSN=EDBC.V2R3.STAGE0.JOB,
//              DISP=(NEW,CATLG,DELETE),
//              UNIT=3380,VOL=SER=volser,
//              SPACE=(TRK,(1,1))
```

2. Make sure the jobcard is valid.
3. For the SYSUT1 DD statement, specify the VOL parameter as listed on the gateway product tape.
4. For the SYSUT2 DD statement, specify:
 - The name under which to store the data set. The default is EDBC.V2R3.STAGE0.JOB.
 - The name of the physical unit on which the gateway data sets will be stored. The default is 3380.
 - The volser number of the disk to which this data set is to be allocated.
5. Submit the job for execution.
6. If you encounter errors restoring this jobstream, correct the errors and repeat until the SYSUT1 file is loaded successfully.

Customizing the Gateway Stage0 Installation Jobstream

The previous step restored the stage0 installation jobstream into a data set with the default name EDBC.V2R3.STAGE0.JOB. The following step requires the gateway product tape as input and creates the gateway installation files as output.

Note: This jobstream assigns a common data set prefix for all installation data sets. The default value for this prefix is `EDBC.V2R3`. It is suggested that you use this default prefix, since it simplifies the installation process. The default data set prefix is used throughout this guide to refer to data set names.

The following table lists the steps executed by the stage0 JCL:

Stepname	Description	RETCODE
STEP010	An IDCAMS step that deletes any previous installation data sets. Returns a non-zero code of 8 if there are no data sets to delete.	0 or 8
STEP020	Executes the ALLOC in-line procedure to allocate the installation data sets.	0
STEP030	Executes the UNLOAD in-line procedure to copy the installation files on the distribution tape to the data sets allocated in STEP020.	0 or 4

Use the ISPF editor to edit the stage0 jobstream as described below, then submit the job. All values that must be customized have the form `#VALUE`.

1. Add a valid jobcard.
2. Customize the following values:
 - **#PREFIX**—Change this string to the value chosen for the installation data set prefix. The prefix can be a 1-n level qualifier. For example, if you choose the default value, `EDBC.V2R3`, issue the ISPF editor command:
`C '#PREFIX' 'EDBC.V2R3' ALL`
 - **#DVOL**—Change this string to the serial number of the volume where the installation data sets will be allocated. For example, if you choose the default value, `MVSVOL`, issue the command:
`C '#DVOL' 'MVSVOL' ALL`
 - **#TAPE**—Change this string to the unit name for the IBM tape drive where the installation product tape is mounted. If you choose the default value, `TAPE`, issue the command:
`C '#TAPE' 'TAPE' ALL`
 - **#TVOL**—Change this string to the serial number of the volume of the installation tape. Refer to the external label of the tape to find the proper value. Issue the command:
`C '#TVOL' 'IVVJ20' ALL`

- **#ADSK**—Change this string to the unit name of the target disk pack on which the data sets will reside. If you choose the default value, 3380, issue the command:

```
C '#ADSK' '3380' ALL
```

- **#TDSK**—Change this string to the unit name of the work disks. If you choose the default value, SYSDA, issue the command:

```
C '#TDSK' 'SYSDA' ALL
```

- **#MAXBLK**—Change this string to the value of the maximum block size for the gateway load libraries. The maximum value is 32760. If you choose the default value, 23476, issue the command:

```
C '#MAXBLK' '23476' ALL
```

3. Submit this job for execution.

STEP010 returns a non-zero return code if there are no data sets to delete. If you encounter any errors, review the output and take corrective action.

When the job completes properly, the gateway installation data sets have been allocated and unloaded from the product tape.

4. Remove the product tape from the tape drive.

Gateway Data Sets That Are Allocated and Restored

The following gateway data sets are allocated by the STAGE0.JOB. These data sets are listed as they would be named with the default prefix:

```
EDBC.V2R3.DOC.TEXT
EDBC.V2R3.SAMPLE.CNTL
EDBC.V2R3.FRONT.LOAD
EDBC.V2R3.BACK.LOAD
EDBC.V2R3.NET.LOAD
EDBC.V2R3.FILES.CLIST
EDBC.V2R3.FILES.MACLIB
EDBC.V2R3.FILES.IIPARM
EDBC.V2R3.FILES.PROCLIB
EDBC.V2R3.FILES.DBRMLIB
EDBC.V2R3.PLI.INCLIB
EDBC.V2R3.TEST.QPLI
EDBC.V2R3.TEST.SPLI
EDBC.V2R3.TEST.PLI
EDBC.V2R3.OBJ
EDBC.V2R3.FILES.H
EDBC.V2R3.FILES.HLP
EDBC.V2R3.FILES.ENGLISH.FAST.MSG
EDBC.V2R3.FILES.ENGLISH.SLOW.MSG
EDBC.V2R3.FILES.NAME.IIAME.ALL
EDBC.V2R3.FILES.USERS.DATA
EDBC.V2R3.FILES.UTEXE.DEF
EDBC.V2R3.WS.LOAD
EDBC.V2R3.WS.CLIB
EDBC.V2R3.WS.H
```

```

EDBC.V2R3.FILES.ASM
EDBC.V2R3.FILES.RTIFORMS.FNX
EDBC.V2R3.FILES.TEMPLATE.ATT.DATA
EDBC.V2R3.FILES.TEMPLATE.IDX.DATA
EDBC.V2R3.FILES.TEMPLATE.REL.DATA
EDBC.V2R3.FILES.TEMPLATE.RELX.DATA
EDBC.V2R3.FILES.NAME.IINAME.INIT
EDBC.V2R3.FILES.ISPLLIB
EDBC.V2R3.FILES.ISPPLIB
EDBC.V2R3.FILES.ISPSLIB
EDBC.V2R3.FILES.SQL
EDBC.V2R3.FILES.SPUFI.IN
EDBC.V2R3.FILES.AMSREPRO.INIT
EDBC.V2R3.STAGES.CNTL
EDBC.V2R3.STAGE2.JOBS
EDBC.V2R3.DFSDDLTO.INIT
EDBC.V2R3.SAMPLE.ASM
EDBC.V2R3.SAMPLE.C
EDBC.V2R3.SAMPLE.COBOL
EDBC.V2R3.SAMPLE.H
EDBC.V2R3.SAMPLE.PLI
EDBC.V2R3.SAMPLE.SASC
EDBC.V2R3.FILES.STARTUP.DATA
EDBC.V2R3.DRIVER.LOAD
EDBC.V2R3.CICS.LOAD
EDBC.V2R3.USER.CNTL
EDBC.V2R3.USER.LOAD
EDBC.V2R3.USER.OBJ
EDBC.V2R3.REPLICAT.LOAD
    
```

Note: The FILES.CLIST data set is allocated with DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160). If these attributes do not conform to your installation standards, use ISPF or some other utility to copy the FILES.CLIST members to a library allocated with the correct DCB attributes.

Customizing the Stage1 Input

The IIVP customizes the Stage1 input, which consists of the statements listed below. The statements are stored in the IGWFSTGT member of the EDBC.V2R3.FILES.ASM data set. These statements use the Assembler H Version 2 syntax. Review each statement, then customize it with ISPF to reflect site-specific values.

Note: Do not modify the macros in EDBC.V2R3.FILES.MACLIB.

The following table lists the stage1 statements and their purpose in the installation:

Statement	Purpose
IGWFJOB	Customizes jobcard parameters
IGWFUSER	Customizes initial gateway users

Statement	Purpose
IGWFINET	Customizes Net
IGWFBVANT	Customizes the Vantage gateway
IGWFPSVR	Customizes the protocol server
IGWFBLD	Builds the IIVP jobstream

Note: The first statement must be IGWFJOB and the last statement must be IGWFBLD. If you specify a parameter with special characters, enclose them in quotes. If you provide a null value, do not place it in quotes.

The following sections define each statement and outline its parameters, usage, and defaults.

IGWFJOB Statement: Specifying Jobcard Parameters

The IGWFJOB statement defines the installation-specific values that are used to build the JCL job statements for the IIVP stage2 jobstreams. You must specify this statement first.

This statement is defined as follows:

```
IGWFJOB  ACCT=' (ACCT, INFO) ',      ACCOUNTING INFO      X
         CLASS=A,                    EXECUTION CLASS      X
         JOBNAME=' IGWFBVP' ,        JOBNAME PREFIX       X
         NOTIFY=,                     NOTIFY USER          X
         MSGCLASS=H,                  MSGCLASS              X
         MSGLEVEL=' (1, 1) ',        MSGLEVEL              X
         PGMNAME=' EDBC INSTALL' ,    PROGRAMMER NAME      X
         REGION=4096K,                REGION SIZE           X
         SYSAFF=,                      SYSTEM AFFINITIES    X
         USER=EDBCDBA                  USER AND EDBC DBA
```

Most of these are standard OS/390 parameters; therefore, enter values that comply with the conventions at your site.

The following table lists only parameters with special significance for the gateway installation procedure:

IGWFJOB Parameter	Usage	Default
ACCT	The information used to build the accounting information in the IIVP jobstream job statements. This information is placed in parentheses on each job statement generated by the IIVP stage1.	''
CLASS	The CLASS parameter on the IIVP jobstream job statements.	A

IGWFJOB Parameter	Usage	Default
JOBNAME	Prefix for each job name in the IIVP stage2 jobstream. The IIVP jobstream appends a two-character suffix to this supplied value. The prefix is limited to a maximum of six characters. If it exceed this limit, the prefix is truncated and a warning message is issued.	IGWFVP
NOTIFY	The value assigned to the NOTIFY parameter of the IIVP job statements.	IGWFJOB USER parameter
MSGCLASS	The MSGCLASS parameter on the IIVP jobstream job statements.	H
MSGLEVEL	The MSGLEVEL parameter on the IIVP jobstream job statements.	(1,1)
PGMRNAME	The programmer name field that appears in each job of the IIVP jobstream. Limited to a maximum of 20 characters. If the value specified exceeds this maximum, it is truncated and a warning message is issued.	'EDBC INSTALL'
REGION	The value assigned to the REGION= parameter of the IIVP JOB STATEMENTS.	0M
USER	Enter the TSO user ID for the gateway DBA. This value is added automatically to the list of users in the IGWFUSER statement. It is also used as the default value for the NOTIFY parameter in this statement.	EDBCDBA
SYSAFF=	Used to generate a /*JOBPARM SYSAFF= statement for generated stage2 jobs. The value specified indicates the systems in a JES2 multi-access spool (MAS) configuration that are eligible to process stage2 jobs. See the <i>JCL Reference</i> guide for acceptable values.	ANY

IGWFUSER Statement: Specifying Initial Gateway Users

The EDBC.V2R3.FILES.USERS.DATA file stores the set of user IDs for remote users who can access the gateway. The TSO user ID for the gateway DBA is automatically added to this list. The file also includes an entry for each gateway subsystem.

The value for the gateway DBA is taken from the value supplied for the USER parameter on the IGWFJOB statement. If the user ID specified for the USER parameter is EDBCDBA, the value EDBCDBA is added to the list.

The IGWFUSER statement specifies the initial users defined to the gateway. The information specified in this statement is used to update data set EDBC.V2R3.FILES.USERS.DATA.

This statement is defined as follows:

```
IGWFUSER  EDCUSR1,    1ST USER DEFINED    X
           EDCUSR2,    2ND USER DEFINED    X
           EDCUSR3     3RD USER DEFINED
```

Each user is specified as a positional parameter. You may specify an unlimited number of users with this technique. It is recommended that the maximum number be limited to 100 users. If you exceed 100, you will run into OS/390 JCL limitations.

IGWFINET Statement: Specifying Net Parameters

The IGWFINET statement specifies the values for initializing and configuring Net.

This statement is defined as follows:

```
IGWFINET  SUBSYS=EDBC,      SUBSYSTEM NAME        X
           ID=II,           INSTALLATION ID        X
           NETLMOD=IIPSERV, NETWORK LOAD MODULE    X
           SVRTYPE=VANT,    DEFAULT SERVER TYPE    X
           SECURITY=NONE,   DEFAULT SECURITY        X
           TIMEOUT=,        GATEWAY WAIT TIMEOUT   X
           MAXUSER=240,     MAXIMUM USERS          X
           TIMEZONE=7       LOCAL TIMEZONE VALUE
```

Customize the parameters for the IGWFINET statement as shown in the following table:

IGWFINET Parameter	Usage	Default
SUBSYS	<p>Four-character identifier for the gateway subsystem on OS/390. This value must be specified.</p> <p>Value must also be defined below, in the IEFSSNxx member in SYS1.PARMLIB.</p> <p>If values are specified for DBNM and ISVR in the previous statement, they must match this value.</p>	EDBC

IGWFINET Parameter	Usage	Default
ID	<p>Two-character installation code for the gateway installation. The value for this parameter must match the value for the INSTALL parameter in the IGWFPSVR statement (see the IGWFPSVR Statement: Specifying Protocol Server Parameters section in this chapter).</p> <p>The installation code is assigned to the following data sets (shown with default names):</p> <p>EDBC.V2R3.FILES.NAME.IICOMSVR.I1 EDBC.V2R3.FILES.NAME.IIDB2.I1 EDBC.V2R3.FILES.NAME.IIIDMS.I1 EDBC.V2R3.FILES.NAME.IIINGRES.I1 EDBC.V2R3.FILES.NAME.IILOGIN.I1 EDBC.V2R3.FILES.NAME.IINODE.I1 EDBC.V2R3.FILES.NAME.IIIMS.I1 EDBC.V2R3.FILES.NAME.IIVSAM.I1 EDBC.V2R3.FILES.NAME.IIVANT.I1</p>	I1
NETLMOD	<p>Name of the network load module to use for Net initialization.</p> <p>The NETLMOD= parameter is used to set the logical symbol, II_NET_LMOD, to the specified value.</p>	IIPSERV
SVRTYPE	<p>Default server class, as specified when accessing Vantage from TSO or a batch application.</p> <p>Note: This value must be entered as shown.</p>	VANT
SECURITY	<p>Type of security package to which the gateway interfaces. The options are:</p> <p>RACF ACF2 TSS None</p> <p>The SECURITY parameter is used to set the logical symbol, II_SECURITY, to the specified value.</p>	NONE
TIMEOUT	<p>Specifies the maximum time, in minutes, that a gateway thread waits before it is disconnected. The range for this parameter is between 0 to 9999. A zero (0) value indicates that no wait time limits are enforced; the Net thread is not disconnected with a 0 value.</p> <p>The TIMEOUT= parameter is used to set the logical symbol, II_INACTV_TMOUTINT, to the specified value.</p>	0
MAXUSER	<p>Maximum number of remote users that can connect to the gateway.</p> <p>Note: This parameter can be set to a value up to 240 for the current release.</p>	64

IGWFINET Parameter	Usage	Default
TIMEZONE	Specifies the difference in hours between Greenwich mean time (GMT) and the local time zone where the gateway server is running.	0

IGWVANT Statement: Specifying Gateway Parameters

The IGWVANT statement defines the:

- Default sizes for the gateway area data set and work sort data set
- Update intent for gateway partitions
- Maximum number of Vantage calls
- The default TCP/IP address and port for the Vantage subsystem
- A default security User ID to be used in connecting to Vantage

This statement is defined as follows:

```
IGWVANT AREARECS=2700,          DEFAULT AREA SIZE          X
        FORMAT=YES,           VANTAGE FORMAT INGAREA    X
        MAXCALLS=999,         MAXIMUM VANTAGE CALLS     X
        SORTRECS=512,        DEFAULT SORT SIZE         X
        TCPIP_ADDR=123.123.123.123,  DEFAULT VANTAGE IP ADR    X
        TCPIP_PORTNUM=1234,     DEFAULT VANTAGE IP PORT    X
        USERID=USERID        DEFAULT USERID FOR VANT CONCT
```

The following table describes each parameter in the IGWVANT statement:

IGWVANT Parameter	Usage	Default
AREARECS	<p>Specifies the default size in 4096-byte records for the gateway area data set.</p> <p>This value is placed in the EDBC Gateway Facility DBA (IGWFDBA) Profile panel. It provides the default size for the skeleton JCL that builds the gateway area.</p> <p>The OS/390 IDCAMS utility rounds this value up to the nearest cylinder boundary when the OS/390 Linear Data set is allocated for the gateway area. The Vantage gateway DBFMT utility formats the primary allocation for this data set.</p>	2700

IGWVANT Parameter	Usage	Default
FORMAT	<p>Specifies whether the gateway area and sort files are to be allocated. The options are:</p> <ul style="list-style-type: none"> ■ NO—Specify NO when the Vantage functions are installed with the EDBC for IMS product. If you specify NO, the gateway area and sort files are assumed to already exist. ■ YES—Specify YES when you are installing the Vantage functions without the EDBC for IMS product. If you specify YES, the gateway area and sort files are destroyed and recreated. 	YES
MAXCALLS	<p>Specifies the maximum number of Vantage calls per SQL transaction.</p> <p>This controls the total work allowed by a Vantage gateway thread before the client must issue a COMMIT</p> <p>If the number of calls exceeds the value set by MAXCALLS, the Vantage gateway thread terminates the last request with an error</p>	999
SORTRECS	Specifies the default size in 4096-byte records for the gateway sort data set.	150
UPDATE	<p>Specifies the update intent for the Vantage gateway. The options are:</p> <ul style="list-style-type: none"> ■ YES—Specifies read/write mode as the default. If you specify YES, all Vantage gateway partitions are built with read/write enabled. ■ NO—Specifies read-only mode as the default. If you specify NO, all Vantage gateway partitions are built with read-only enabled. 	YES
TCPIP_ADDR	data setThe default TCP/IP address for an active Vantage subsystem. This parameter is used to generate stage2 JCL that can connect to an active Vantage subsystem to enumerate and describe Vantage objects as EDBC partition tables. If you are licensed for and running more than one Vantage subsystem, you can use this stage2 JCL as a model to create additional jobs/data sets to discover and describe other Vantage subsystems to EDBC.	None
TCPIP_PORTNUM	The default TCP/IP port number for an active Vantage subsystem as described by the TCPIP_ADDR parameter (see the TCPIP_ADDR parameter for more information on this parameter).	None

IGWVANT Parameter	Usage	Default
USERID	Vantage normally requires that EDBC provide a security User ID and Password when EDBC connects to an active Vantage subsystem. This is to ensure the integrity of the TCP/IP interface to Vantage. This identifies the default User ID to be used in the stage2 job that connects to the Vantage subsystem identified by the TCPIP_ADDR and TCPIP_PORTNUM parameters. The EDBC utility program executed by that stage2 job will prompt you for the password via a WTOR at runtime.	None

IGWFPSVR Statement: Specifying Protocol Server Parameters

The IGWFPSVR statement specifies the network parameters for the Net protocol servers, CCI, SNA LU0, SNA LU62, IBM TCP/IP, SNS/TCP, and KNET TCP/IP. This statement is coded, assembled, and linked into load module form during the installation process. The jobstream can be built and saved to allow addition of new entries.

Note: For SNA LU0, notify the client System Programmer of the value for the DLOGMOD parameter, and the ACBNAME parameter, which defines the name of the minor node for the SNA LU0 application.

For SNA LU62, notify the client System Programmer of the value for the DLOGMOD parameter and the minor node name that corresponds to the ACBNAME parameter for the SNA LU62 application.

For KNET TCP/IP, notify the client System Programmer of the value of the port ID.

For all TCP/IP protocol servers, verify that the port ID is the same on the client and the Net server.

For CCI, verify that the Product ID is unique for the particular CCI system that the Net server is associated with.

The IGWFPSVR statement can be repeated up to 32 times for each protocol server type.

The IGWFPSVR statement is defined as follows:

```

IGWFPSVR  TYPE=SNA_LU0,          TYPE OF SERVER          X
          MODETAB='RTIMODE',    MODE TABLE             X
          DLOGMOD='RTILU0',     DEFAULT LOGON MODE     X
          ACB='RTIACB11',      ACB NAME               X
          RUSIZE=4096,         MAXIMUM RU SIZE       X
          PLU=,                 KNET PLU NAME         X
          PORT='134',         PORT ADDRESS          X
          INSTALL=11,        INSTALLATION CODE     X
          PASS=' ',         OPTIONAL PASSWORD     X
          NODENAME=,         LOGICAL UNIT NAME     X
          USERID='TCP/IP',    IBM TCP/IP USERID    X
          APPLID=,          APPLICATION ID - SNS/TCP X
          SYSID=,          SNS/TCP SUBSYSTEM ID  X
          PRODID=,        CCI SUBSYSTEM ID
    
```

Customize the parameters for the IGWFPSVR statement as shown in the following table. If you do not enter a value for a parameter, it defaults to the value shown in the table. The system ignores values for non-selected protocols.

IGWFPSVR Parameter	Usage	Default
TYPE=	Type of protocol server. The options are: <ul style="list-style-type: none"> ■ <i>SNA_LU0</i> for SNA LU0 ■ <i>SNA_LU62</i> for SNA LU6.2 ■ <i>TCP_KNET</i> for KNET TCP/IP ■ <i>TCP_IBM</i> for IBM TCP/IP ■ <i>TCP_SNS</i> for SNS/TCP ■ <i>CCI</i> for CAI CCI 	<i>SNA_LU0</i>
MODETAB=	SNA LU0 or SNA LU6.2 only: Name of the VTAM mode table that contains the default logon mode entry for this protocol server. Used to build the APPL statement and create the VTAM logon mode table for this protocol server.	<i>EDCMODE</i>
DLOGMOD=	SNA LU0 or SNA LU6.2 only: In conjunction with the MODETAB= parameter this parameter defines the default logon mode to be used in establishing sessions with this application.	<i>EDCLU0</i>

IGWFPSVR Parameter	Usage	Default
ACB=	<p>SNA LU0 or SNA LU6.2 only:</p> <p>Name of the VTAM Access Control Block (ACB) used by the Net protocol servers. Also used to create the APPL definition for SNA protocol servers. This value is required to initialize the protocol interfaces.</p> <p>The value should not exceed eight characters. It must begin with an alpha or national character.</p>	<i>EDCACB11</i>
RUSIZE=	<p>Sets the maximum RU size for the protocol servers.</p> <p>This value should be consistent with your installation's SNA standards. The default value is recommended.</p>	4096
PLU=	<p>KNET TCP/IP only:</p> <p>This value is the VTAM ACBNAME used by the KNET address space. The Net server must have access to the KNET runtime library either through a STEPLIB or LNKLST specification.</p>	None
PORT=	<p>All TCP/IP protocols:</p> <p>Value assigned for the port ID that is used to accept connections from remote EDBC users.</p> <p>The port ID is an arbitrary value. Do not set the port ID equal to 7, 9, 11, 13, 17, 19, 20, 21, 23, 25, or 69. These values are used by KNET. Additionally, other services have assigned port numbers for use by TCP and UDP (User Datagram Protocol). Generally, avoid numbers lower than 1024 as these tend to be reserved for well-known TCP and UDP services.</p> <p>The exception to this rule is the default of 134, which is officially reserved for Net and should cause no conflicts. If you have more than one Net installation within the same OS/390 system, specify a unique port ID for each installation.</p>	134

IGWFPSVR Parameter	Usage	Default
INSTALL=	Two-character installation code for the gateway. If unspecified, defaults to the value assigned to the IGWFINET ID= parameter (see the IGWFINET Statement: Specifying Net Parameters section in this chapter).	I1
PASS=	Password.	None
NODENAME=	SNA_LU0 and SNA_LU62 only: Specifies the VTAM logical unit name for the associated ACB name. This parameter allows the LU name to differ from the ACB name. If NODENAME= is not specified, it defaults to the value of the ACB= parameter. If NODENAME= is specified and ACB= is not specified, ACB= defaults to the NODENAME= value. This parameter provides the label for the VTAM APPL statement generated by the IIVP stage2 jobstream.	ACB= parameter
USERID=	IBM TCP/IP only: This is the user ID of the IBM TCP/IP address space. The statement default value is TCPIP, which is also the default when installing IBM TCP/IP. Check with the system programmer to determine whether a value other than this default was used.	TCPIP

IGWFPSVR Parameter	Usage	Default
ENABLE=	<p>Specifies whether the IGWFPSVR statement will be processed or ignored during stage1. This parameter allows all protocol servers to be defined without enabling them. The options are:</p> <ul style="list-style-type: none"> ■ NO – IGWFPSVR statement is processed but the protocol server is not enabled during the gateway initialization. During stage1, results in a return code of 04. ■ YES – IGWFPSVR statement is processed. The protocol server is enabled during the gateway initialization. <p>All protocol initialization parameters are placed in one load module in EDBC.V2R3.NET.LOAD (IIPSERV). A member that can recreate this load module is customized in the IIVP stage2 jobstream.</p> <p>It is located in EDBC.V2R3.SAMPLE.CNTL(ASMPSERV).</p>	YES
APPLID=	<p>SNS/TCP only:</p> <p>Specifies the application program name that is passed to SNS/TCP. This name is used with the PASSWORD= value to authorize access to SNS/TCP.</p> <p>The name, if coded, must be an alphanumeric string up to 8 characters long.</p>	Nulls
SYSID=	<p>SNS/TCP only:</p> <p>Specifies the name of the SNS/TCP subsystem that is to be invoked by the TCP_SNS protocol server.</p> <p>The subsystem name is an alphanumeric string up to 4 characters long.</p>	ACSS
PRODID=	<p>CCI only:</p> <p>Applies only to the CCI system that is running. It must be a unique 20 character maximum Product ID and unique within the CCI system.</p>	

IGWFBLD Statement: Specifying IIVP Sysgen Parameters

The IGWFBLD statement specifies global default IIVP parameters and builds the IIVP stage2 jobstream.

Note: This statement must be last in the stage1 input.

The IGWFBLD statement is defined as follows:

```
IGWFBLD RELEASE='EA VANT 2.3/0112 (MVS.390/00)', EDBC RELEASE      X
        AMODGEN='SYS1.MACLIB', INPUT AMODGEN MACLIB             X
        ASMBLR='IEV90', IBM ASSEMBLER                           X
        LINKLIB=, DEFAULT LINKLIB                                X
        LIST=NO, DON'T SHOW LISTING                             X
        PROCLIB=, DEFAULT PROCLIB                               X
        EDBC='EDBC.V2R3', EDBC DATA SET PREFIX                 X
        SORTLIB=, DEFAULT SORTLIB                               X
        STORCLAS=, SMS STORAGE CLASS                             X
        SYSDA='SYSDA', DEFAULT WORK UNIT                        X
        TYPE=GEN, GENERATE JOBSTREAM                             X
        UNIT=, DEFAULT UNITNAME                                  X
        VTAMLST=, DEFAULT VTAMLST                               X
        VTAMMAC='SYS1.MACLIB', INPUT AMODGEN MACLIB             X
        VOLSER=MVSVOL, EDBC DEFAULT VOLSER                     X
        PRODUCTS=(VANTGW,EDBC) SPECIFY EDBC PRODUCTS
```

Customize the parameters for the IGWFBLD statement as shown in the following table:

IGWFBLD Parameter	Usage	Default
RELEASE	Release level of the gateway installation. Note: This value must be entered as defined in the Release Notes with the product tape.	None
AMODGEN	Name of system macro library.	SYS1. AMODGEN
ASMBLR	Specifies which IBM ASSEMBLER is to be used by the stage2 sysgen. Valid values are IEV90 and ASMA90	ASMA90
LINKLIB	Names the data set in the LNKLSTxx member of SYS1.PARMLIB into which the IGWF subsystem initialization modules will be copied. If you intend to add EDBC.V2R3.BACK.LOAD to the linklist, you may omit this parameter and let it default.	EDBC . VANTGW. BACK. LOAD

IGWFBLD Parameter	Usage	Default
LIST	<p>Specifies whether the output of the IIVP stage1 should be a summary or a detail listing:</p> <ul style="list-style-type: none"> ■ NO—Specifies only a summary listing, describing each parameter and its default. ■ YES—Specifies an assembler listing showing how each statement expanded to produce the stage2 jobstream. Used for diagnosing errors in the IIVP stage1 generation process. 	NO
PROCLIB	Name of the OS/390 PROCLIB that is updated with cataloged procedures that are specific to the gateway.	EDBC VANTGW. FILES. PROCLIB
PREFIX	<p>Data set prefix for the gateway installation data sets. You must create an OS/390 alias for the high-level index of the data sets if one does not already exist.</p> <p>The value for this parameter must match the #PREFIX value as defined in the EDBC.V2R3.STAGE0.JOB.</p>	EDBC VANTGW
SORTLIB	Name of SORTLIB used by the stage2 sysgen.	SYS1. SORTLIB
STORCLAS	<p>Specifies the SMS storage class to use for the allocation of the name server files.</p> <p>This value overrides the UNIT= and VOLSER= specifications.</p>	UNIT= VOLSER=
SYSDA	OS/390 unit name used for workspace allocation	SYSDA
TYPE	<p>Type of IIVP sysgen:</p> <ul style="list-style-type: none"> ■ GEN—The IIVP stage1 should produce the stage2 jobstream if the return code is no higher than 4. If the return code is higher than 4, some default was taken that requires review. ■ NOGEN—The IIVP stage1 should not produce the stage2 jobstream. 	GEN

IGWFBLD Parameter	Usage	Default
UNIT	Name of the physical device where the name server files will be allocated.	3380
VTAMLIB	SNA LU0 or SNA LU62 only: Name of the VTAM link library that is the target for the mode table for the application used by the SNA protocol servers.	EDBC. VANTGW. NET. LOAD
VTAMLST	Name of the VTAMLST data set that is updated with the VTAM application node for the designated protocol server.	EDBC VANTAGEGW SAMPLE. CNTL
VTAMMAC	Name of VTAM macro library. For MVS/ESA, specify SYS1.MACLIB. For OS/390, specify SYS1.SISTMAC1	SYS1. MACLIB
VOLSER	Serial number of the volume onto which the Vantage gateway data sets will be allocated.	MVSVOL
PRODUCTS	Products that are to be customized for installation, in parentheses, separated by a comma. Note: The only valid values are VANTGW and EDBC. You must explicitly enter the values; they are not listed by default.	None

The Stage1 Jobstream

The stage1 jobstream uses the stage1 input, EDBC.V2R3.FILES.ASM (IGWFSTGT) to produce the stage2 jobstream, a report, and a listing.

To do this you need to customize the stage1 jobstream in the data set with the default name EDBC.V2R3.SAMPLE.CNTL (IGWFSTGT). The following section details this procedure.

Customizing and Executing the IGWFSTGT JCL

The following procedure customizes and executes the IGWFSTGT JCL:

1. Use the ISPF editor to edit the member IGWFSTGT in data set EDBC.V2R3.SAMPLE.CNTL:
 - Add an installation job statement.

- On the STAGE1 EXEC statement, customize the EDBC parameter by changing the 'EDBC.VxRy' string to the value of the gateway data set prefix. If the value chosen is EDBC.V2R3, then use the following ISPF editor command to customize this value:

```
C 'EDBC.VxRy' 'EDBC.V2R3' ALL
```

The name of the library AMODGEN may be different, depending on the version of OS/390 you are running.

2. Submit the job for execution.

The job gets a return code of 0 or 4. It produces:

- The IIVP jobstream. The following jobs are created in data set EDBC.V2R3.STAGE2.CNTL:

```
VANTA0
VANTA1
VANTB0
VANTB1
VANTIO
VANTN0
VANTT0
VANTT1
VANTT2
VANTT3
VANTZ9
```

- SYSPRINT stage1 Report. This file contains a summary of all parameters and defaults selected. It shows the products that have been selected for installation by the IIVP jobstream and lists the jobs that were created in the IIVP jobstream.
- SYSLIN IIVP Jobstream Listing. This file contains a listing of the IIVP jobstream. This listing is provided for diagnostic and documentation purposes.

Submitting the Stage2 Jobstream Jobs

Use the following procedure to edit and release each of the jobs in the IIVP stage2 jobstream. This section summarizes the overall procedure. The sections that follow describe each job in detail.

1. Edit the jobs as required following site standards to add jobcards, passwords, and user IDs.

- Submit the jobs one at a time in the exact sequence specified in the following list:

VANTA0 Job	Customizes the logical symbols jobstream
VANTA1 Job	Allocates registration scripts data set
VANTB0 Job	Installs the IGWF subsystem dynamically. If you prefer, skip this job and follow the instructions in this chapter's Installing the IGWF Subsystem Permanently section.
VANTB2 Job	Customizes the IGWFDBA screens
VANTI0 Job	Customizes Net parameters
VANTN0 Job	Builds the Name server files
VANTT0 Job	Create mvs area and sort files
VANTT1 Job	Create Vantage partition
VANTT2 Job	Generate Vantage registration scripts
VANTT3 Job	Vantage batch test job/IVP
VANTZ9 Job	JCL to start the EDBC server

- Check the completion status of each job after it is released.
- If you encounter an error, make the necessary corrections and resubmit the failed job.

Note: You can restart each job in the IIVP stage2 jobstream on a job boundary. If you need to return to the stage1 step, you can do so without any harm. The stage2 jobs delete any data sets that were previously created.

Functions of Stage2 Jobs

The following sections describe each job in the IIVP stage2 jobstream.

IGWVPA1 Job: Allocating the Register File

This job allocates a register file. The IMS D1 job uses this file to store the auto-registration script. The Vantage T2 job uses this file to store its registration scripts. For VSAM, this file is unused at present.

This job should only be run once unless there is an allocation error that needs to be corrected.

Stepname	Description	RETCODE
STEP010	Delete IMS file	0
STEP020	Allocate IMS file	0

Stepname	Description	RETCODE
STEP030	Allocate IMS file	0
STEP040	Allocate VSAM file	0
STEP050	Delete Vantage file	0
STEP060	Allocate Vantage file	0

Libraries Updated by
IGWFVPA1 Job

The libraries affected by this job are:

EDBC.V2R3.IMS.REGISTER
EDBC.V2R3.VANTAGE.REGISTER
EDBC.V2R3.VSAM.REGISTER

IGWFVPB0 Job:
Installing the IGWF
Subsystem

The following table lists the steps executed by the IGWFVPB0 job:

Stepname	Description	RETCODE
STEP010	Copies IGWF modules in LINKLIB if the LINKLIB= parameter was specified in the IGWFBLD macro. The dataset specified or defaulted to MUST be cataloged in the MVS MASTER catalog. If it is not, the IGWF subsystem will not initialize properly.	0
STEP015	Installs the IGWF subsystem. If this step fails, analyze messages and take appropriate action as indicated by the return code. Take corrective action and install the IGWF subsystem before proceeding.	0-8

The following table describes the return codes for STEP015:

Stepname	Description and Action
0	IGFW subsystem installation successful. Proceed to the next step.
1	IGFW subsystem already built. The IGWF subsystem has already been initialized on your OS/390 system. The initialization module determined that your OS/390 system already contains an IGWF subsystem. Re-run STEP015 with PARM=REINSTALL.
2	Memory allocation or load of function code failed. The initialization module determined that the subsystem function routines could not be loaded. Check that your system contains these modules: GWFFC00 GWFFC01 GWFFC02 GWFFC03 GWFFC08 GWFFC10 GWFFCI0 GWFFCI1 Check that your OS/390 system has at least 20 KB of CSA free below the 16 MB line.
3	Memory allocation for IGWF SSCT failed. The initialization module determined that there was insufficient memory to initialize the IGWF subsystem communications table. Check that your OS/390 system has at least 20 KB of CSA free below the 16 MB line.
4	Memory allocation for PC hold area failed. The initialization module determined that there was not enough memory to initialize the IGWF program call routine storage. Check that your OS/390 system has at least 20 KB of CSA free below the 16 MB line.

Stepname	Description and Action
5	<p>Memory allocation or load of program call routine failed.</p> <p>The initialization module determined that the subsystem program call or program call routines could not be loaded.</p> <p>Check that the IGWF subsystem library contains the following modules:</p> <p>GWFPC00 GWFPC01 DD@PC22 DD@PC31 GWFSCHK DD@PC41 DD@PC51 DD@PC61 DD@PC71 IGWFAG</p> <p>Check that your OS/390 system has at least 20 KB of CSA free below the 16 MB line.</p>
6	<p>Memory allocation for the IGWF SSVT failed.</p> <p>The initialization module determined that there was insufficient memory to initialize the IGWF subsystem vector table.</p> <p>Check that your OS/390 system has at least 20 KB of CSA free below the 16 MB line.</p>
7	<p>Number of subsystems does not match count in JESCT.</p> <p>The initialization module determined that the subsystem chain does not match the current subsystem hash table. This occurs when vendor software builds OS/390 subsystems dynamically and fails to update the OS/390 subsystem hash table.</p> <p>The hash table is created and initialized during normal IPL of the OS/390 system and is used by the master subsystem to locate subsystems efficiently. STEP015 determines if the hash table matches the subsystem chain before updating the subsystem chain. If the hash table has not been properly maintained by other vendor software, STEP015 returns this status.</p> <p>If you receive this return code, you must install the IGWF subsystem manually. See this chapter's section entitled, Installing the IGWF Subsystem Permanently for more information.</p>

Stepname	Description and Action
8	IGWFSSCT not found. An attempt was made to run IGWFSSD with PARM=REINSTALL but IGWF had not already been installed. Re-run IGWFSSD without PARM=REINSTALL.
STEP020	Maps the IGWF subsystem modules. Displays the load point and maintenance levels of the IGWF subsystem modules.
STEP050	Deletes old registration scripts data set - rc = 0 or 4
STEP060	Allocates the registration scripts data set - rc = 0

Libraries Updated by IGWFVPB0 Job

The libraries updated by the IGWFVPB0 job depend on the values specified in the IIVP stage1 input file.

- If IGWFBLD LINKLIB contains the name of an OS/390 LNKLST library, then that library is updated with the IGWF subsystem modules.
- If IGWFBLD LINKLIB is allowed to default, or if it is the same as EDBC.V2R3.BACK.LOAD, then no libraries are updated.

STEP015 assumes that the EDBC load library, EDBC.V2R3.BACK.LOAD, has been APF-authorized. This library will be referenced when the IGWF subsystem is updated.

IGWFVPB2 Job:
Customizing the Gateway Facility DBA ISPF Panels

This job customizes the EDBC Gateway Facility Database Administrator (IGWFDDBA) skeleton and profile initialization variables for the IGWFDDBA utility. The following table lists the step executed by the IGWFVPB2 job:

Stepname	Description	RETCODE
STEP190	Customizes IGWFDDBA ISPPLIB Dataset	0

The following table lists the library updated by the IGWFVPB2 job:

Dataset Updated	Description of Dataset
EDBC.V2R3.FILES.ISPPLIB	IGWFDDBA ISPF panels library

The information in this library is updated with the installation values obtained from the IIVP stage1. No further customization of the IGWFDDBA utility is required in order to use the product.

VANTA0 Job:
Customizing the
Logical Symbols
Jobstream

This job customizes the installation and run-time parameters that are called logical symbols in the gateway. The following table lists the steps executed by the IGWFPA0 job:

Stepname	Description	RETCODE
STEP010	Adds members to IIPARM PDS	0
STEPU1nn	Adds user entries to USERS.DATA file	0 or 4
STEPS1nn	Adds subsystem entries to USERS.DATA file	0 or 4

The following table lists the libraries updated by the VANTA0 job:

Data Sets Updated	Description of Data Set
EDBC.V2R3.FILES.IIPARM	Logical symbol file
EDBC.V2R3.FILES.USERS.DATA	EDBC authorization file

VANTI0 Job:
Customizing Net
Parameters

This job customizes the Net parameters. If system libraries are not the targets of the steps in this process, you must copy the resulting files to the appropriate libraries before you can initialize Net.

The following table lists the steps executed by the VANTI0 job:

Stepname	Description	RETCODE
STEP010	Assembles the Network Load Module	0
STEP015	Link edits the Network Load Module	0
STEP017	Creates the ASMPSERV JCL Member	0
STEP020	Creates APPL definition for LU0	0
STEP030	Creates in-line JCL for Net	0
STEP040	Assembles Net Mode Table	0 or 8
STEP050	Link edits Net Mode Table	0 or 8
STEP060	Creates Net procedure	0
STEP070	Batch JCL for Net	0
STEP080	Creates Post Installation in JCL	0
STEP090	Creates IIPARM CLIST	0

The following table lists the libraries updated by the VANTI0 job:

Data Sets Updated	Comments
EDBC.V2R3.NET.LOAD	or IGWFBLD VTAMLIB
EDBC.V2R3.SAMPLE.CNTL	or IGWFBLD VTAMLST
EDBC.V2R3.FILES.PROCLIB	or IGWFBLD PROCLIB

VANTN0 JOB: Create Name Server Files The VANTN0 job creates the data sets required by the name server component of the OS/390 Net server.

Stepname	Description	RETCODE																								
STEP010	Deletes the existing name server data sets	0																								
STEP020	Allocates the name server data sets as follows:	0																								
	<table> <thead> <tr> <th>Data set</th> <th>Size</th> </tr> </thead> <tbody> <tr> <td>-----</td> <td></td> </tr> <tr> <td><i>prefix.FILES.NAME.IICOMSRV.id</i></td> <td>1 Trk</td> </tr> <tr> <td><i>prefix.FILES.NAME.IIINGRES.id</i></td> <td>1 Trk</td> </tr> <tr> <td><i>prefix.FILES.NAME.IILOGIN.id</i></td> <td>1 Trk</td> </tr> <tr> <td><i>prefix.FILES.NAME.IINODE.id</i></td> <td>1 Trk</td> </tr> <tr> <td><i>prefix.FILES.NAME.IIDB2.id</i></td> <td>1 Trk</td> </tr> <tr> <td><i>prefix.FILES.NAME.IIDCOM.id</i></td> <td>1 Trk</td> </tr> <tr> <td><i>prefix.FILES.NAME.IIIDMS.id</i></td> <td>1 Trk</td> </tr> <tr> <td><i>prefix.FILES.NAME.IIIMS.id</i></td> <td>1 Trk</td> </tr> <tr> <td><i>prefix.FILES.NAME.IIVSAM.id</i></td> <td>1 Trk</td> </tr> <tr> <td><i>prefix.FILES.NAME.IIVANT.id</i></td> <td>1 Trk</td> </tr> </tbody> </table>	Data set	Size	-----		<i>prefix.FILES.NAME.IICOMSRV.id</i>	1 Trk	<i>prefix.FILES.NAME.IIINGRES.id</i>	1 Trk	<i>prefix.FILES.NAME.IILOGIN.id</i>	1 Trk	<i>prefix.FILES.NAME.IINODE.id</i>	1 Trk	<i>prefix.FILES.NAME.IIDB2.id</i>	1 Trk	<i>prefix.FILES.NAME.IIDCOM.id</i>	1 Trk	<i>prefix.FILES.NAME.IIIDMS.id</i>	1 Trk	<i>prefix.FILES.NAME.IIIMS.id</i>	1 Trk	<i>prefix.FILES.NAME.IIVSAM.id</i>	1 Trk	<i>prefix.FILES.NAME.IIVANT.id</i>	1 Trk	
Data set	Size																									

<i>prefix.FILES.NAME.IICOMSRV.id</i>	1 Trk																									
<i>prefix.FILES.NAME.IIINGRES.id</i>	1 Trk																									
<i>prefix.FILES.NAME.IILOGIN.id</i>	1 Trk																									
<i>prefix.FILES.NAME.IINODE.id</i>	1 Trk																									
<i>prefix.FILES.NAME.IIDB2.id</i>	1 Trk																									
<i>prefix.FILES.NAME.IIDCOM.id</i>	1 Trk																									
<i>prefix.FILES.NAME.IIIDMS.id</i>	1 Trk																									
<i>prefix.FILES.NAME.IIIMS.id</i>	1 Trk																									
<i>prefix.FILES.NAME.IIVSAM.id</i>	1 Trk																									
<i>prefix.FILES.NAME.IIVANT.id</i>	1 Trk																									
	Where:																									
	<i>prefix</i> is value of PREFIX= parameter																									
	<i>id</i> is the two character installation code for the gateway installation																									
STEP030	Initializes the <i>prefix.FILES.NAME.IIDB2.id</i>	0																								
STEP040	Initializes the <i>prefix.FILES.NAME.IIIMS.id</i>	0																								
STEP050	Initializes the <i>prefix.FILES.NAME.IIVSAM.id</i>	0																								
STEP060	Initializes the <i>prefix.FILES.NAME.IIDCOM.id</i>	0																								
STEP070	Initializes the <i>prefix.FILES.NAME.IIIDM.id</i>	0																								
STEP080	Initializes the <i>prefix.FILES.NAME.IIVANT.id</i>	0																								

Stepname	Description	RETCODE
STEP010	Delete the scripts file prefix.VANTAGE.REGISTER	4
STEP020	Allocate the scripts file prefix.VANTAGE.REGISTER	0
VANTEDBC	Execute the VANTEDBC utility program, which connects to Vantage, enumerating the Vantage objects and creating EDBC registration scripts for each Vantage object. Note: Before going on to the VANTT3 stage2 job, you should use TSO to edit the \$RUNJOB member of the prefix.VANTAGE.REGISTER file and follow the instructions for customizing this JCL. Once you have customized the JCL, submit this job and allow it to run before running the VANTT3 job.	0

VANTT3 JOB: Verify install

The VANTT3 job performs an SQL query that involves getting data from Vantage. If the Vantage gateway has been installed properly, the job should run without errors.

Stepname	Description	RETCODE
STEP010	Execute the SQLDBA batch program to execute an SQL query against the Vantage partition.	4

Final Installation Procedures

This section outlines the remaining installation procedures.

Customizing the TSO Logon Procedure

At this point the Vantage gateway has been installed and customized, and the sample databases have been loaded. Now you must modify the TSO logon procedure for the gateway DBA. These modifications enable you to access the IGWFDBA utilities that are used for gateway checkout, operation, and maintenance.

To customize your TSO logon procedure, add the following DD statements to the TSO logon procedure, or place them in the DD concatenation sequence if the DD statement already exists. The data sets are shown here with the default names:

```
//STEPLIB DD DISP=SHR,DSN=EDBC.V2R3.FRONT.LOAD
// DD DISP=SHR,DSN=EDBC.V2R3.BACK.LOAD
// DD DISP=SHR,DSN=EDBC.V2R3.NET.LOAD
.
//SYSPROC DISP SHR,DSN=EDBC.V2R3.FILES.CLIST
.
//ISPLLIB DISP=SHR,DSN=EDBC.V2R3.FILES.ISPLLIB
.
//ISPPLIB DISP=SHR,DSN=EDBC.V2R3.FILES.ISPPLIB
.
//ISPSLIB DISP=SHR,DSN=EDBC.V2R3.FILES.ISPSLIB
```

Adding the Gateway Data Sets to the APF List

To allow the gateway to initialize successfully, add the following data sets to the IEAAPFxx member in SYS1.PARMLIB. The default names and parameters for these data sets are:

```
EDBC.V2R3.BACK.LOAD VOLSER,
```

```
EDBC.V2R3.NET.LOAD VOLSER,
```

```
EDBC.V2R3.DRIVER.LOAD VOLSER,
```

The value for the VOLSER must match the value specified on the VOLSER parameter in the IGWFBLD statement. The entry for EDBC.V2R3.DRIVER.LOAD is required only if the library will be concatenated to a CICS steplib; see the section OS/390 Interface-Accessed Programs in the “Setting Up Vantage Drivers” chapter.

Performing an IPL of the OS/390 System

If you installed the IGWF subsystem dynamically, you can test and work with the gateway before an IPL is performed. As gateway DBA, you now have access to the gateway locally, from a batch job or TSO address space. Remote users cannot access the gateway until the gateway server is started.

If you installed the IGWF subsystem permanently, you must IPL the OS/390 system before you can go on to verify the gateway’s functionality.

If you elected to install the IGWF subsystem permanently, you should verify the integrity of the subsystem after the next IPL by submitting &prefix.SAMPLE.CNTL(IGWFZAP) and analyzing the output for errors. Here is an example of an IGWFZAP INSPECT on a system with IGWF correctly installed:

IGWFZAP12: Storage map of IGWF subsystem
Copyright(c), 1996,2001 Computer Associates International, Inc.

Address	Module	Maintenance Level
00C6C7B8	IGWF SSCT	
80C5F3B8	GWFFC00	GWFFC00-88287-01/29/98-23.52
80C5D368	GWFFC01	GWFFC01-81840-10/09/97-18.51
80C85000	GWFSCHK	GWFSCHK-BASE-06/18/96-21.59
80C66050	DD@PC22	UNKNOWN
80C5E560	DD@PC31	DD@PC31-310-BASE-11/11/95-02.47
80C5C0A0	DD@PC41	DD@PC41-410-99711-2-05/03/00-19.55
80C5B070	DD@PC51	DD@PC51-510-99711-2-05/03/00-16.20
80C5A070	DD@PC61	DD@PC61-610-99711-2-05/03/00-16.22
80A0E000	DD@PC71	DD@PC71-710-103100-11/05/00-19.21
80C6A330	IGWFAG	IGWFAG
00C645A0	GWFFC00	GWFFC00-88287-02/01/98-19.57
00C63528	GWFFC01	GWFFC01-88287-02/01/98-19.48
00C85040	GWFFC02	GWFFC02-77063-06/18/96-21.57
00C73030	GWFFC03	GWFFC03-77063-06/18/96-21.57
00C627A0	GWFFC08	GWFFC08-87927-01/02/98-16.26
00C6A1D0	GWFFC10	GWFFC10-77063-06/18/96-21.58
00C61558	GWFFCI0	GWFFCI0-83535-06/30/97-16.03
00C60258	GWFFCI1	GWFFCI1-83535-06/30/97-22.26

Any abend in IGWFZAP INSPECT or a maintenance level of "UNKNOWN" for any module other than DD@PC22 is an indication of a corrupted IGWF subsystem. See Diagnosing IGWF Initialization Failures After an IPL in the Debugging the Gateway section in the "Maintaining the Gateway" chapter. Rerunning the B0 will repair the subsystem until the next IPL.

Define the EDBC server subsystems to OS/390 so that the subsystem is initialized after each IPL. Do so by adding the following entries to the IEFSSNxx member in SYS1.PARMLIB:

For the EDBC server subsystem, the name must correspond to the value specified in the SUBSYS parameter of the IGWFINET statement, defined earlier in this chapter. The default value is EDBC. A sample entry is:

EDBC for Vantage gateway

The Next Step

The gateway and most of the EDBC server are now installed. You have two options at this point:

- You can log on from TSO to verify that you have local access to the gateway. Using the gateway utilities and the Terminal Monitor, you can run SQL queries against the sample databases created during installation. This allows you to verify that the gateway is installed successfully. The “Verifying the Gateway” installation chapter describes this process.
- Start the EDBC server and connect to EDBC from remote clients.

Verifying the Gateway

To verify that the gateway has been installed and configured properly, it is necessary to access Vantage data through the gateway.

Any Vantage query through the gateway verifies whether the gateway has been installed and configured properly. However, a specific stage2 job (T3) is created during the install process that performs a very simple query using the SQLDBA batch utility.

This chapter describes what the gateway install process establishes and how to make use of the verification job provided with the stage2 JCL.

Gateway Verification Summary

The installation process automatically creates the following resources for you as the gateway DBA:

- A gateway area with a sort work data set
- The logical symbols that enable you to access this area
- A partition that supports access to an initial VANTAGE subsystem
- A utility and process to help you create the partition table registrations to fully and accurately reflect the record layouts of Vantage objects

To verify the gateway, simply submit the T3 stage2 job and verify that there were no fatal errors. This job should display each of the objects defined to Vantage along with a description of these objects.

The Next Step

Once you have verified that the gateway has been properly installed, you can finish customizing the IBM networking software and start up the entire gateway address space. The “Configuring the EDBC Server” chapter describes this process.

Using the IGWFDBA Utilities

This chapter describes how to use the EDBC Gateway Facility Database Administrator (IGWFDBA) utilities. The utilities consist of a series of ISPF panels that automate the creation of skeleton JCL to perform gateway administrative operations. The IGWFDBA utilities should reside in a non-APF authorized library. Only authorized personnel should have access to the IGWFDBA utilities.

The IGWFDBA utilities allow the gateway DBA to do the following operations:

- Test the gateway and verify its successful installation, as described in the “Verifying the Gateway Installation” chapter.
- Set up gateway users and create the logical symbols, areas, and partitions as described in the “Setting Up Vantage Gateway Users” chapter.
- Maintain the gateway, back up and restore gateway areas, and carry out other maintenance tasks, as described in the “Maintaining the Gateway” chapter.

IGWFDBA Main Menu

The IGWFDBA Main Menu allows you to access the IGWFDBA utilities through a series of ISPF panels. The panels provide options for you to maintain, access, and support the Vantage gateway (and, if applicable, the VSAM, IMS or DB2 gateway).

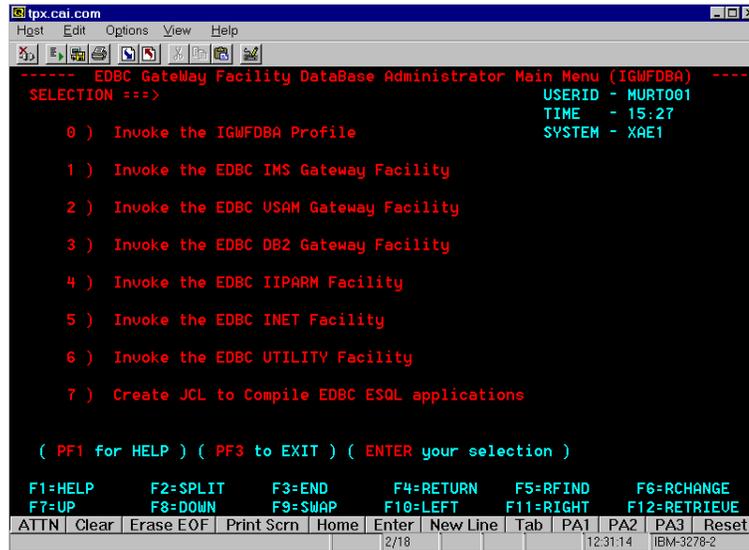
Using This Menu

To invoke the IGWFDBA Main Menu:

1. Log on to TSO as the gateway DBA using the revised TSO logon procedure.
2. In ISPF, from the Primary Option menu, select Option 6, the TSO command.
3. Type **IGWFDBA** and press Enter.
4. The terminal displays a panel similar to the following screen. The displayed options are described in the Main Menu Options section.

5. Type the number of the selection you want to access in the Selection field and press Enter.
6. The terminal displays the panel that corresponds to your entry.
7. The procedures in this chapter describe how to use Options 0, 2, 4, 5, 6 and 7. For more information about your selection go to the section in this chapter that describes it.

To return to the TSO command panel, press PF3.



Main Menu Options

The following options are available from the IGWFDBA Main Menu:

Field	Description
Option 0	The profile set up panel. This enables you to predefine a series of variables that are used by the IGWFDBA utilities.
Option 1	The IMS gateway main menu. For more information on these panels, see the EDBC for IMS documentation.
Option 2	The VSAM gateway main menu.
Option 3	The DB2 gateway main menu.
Option 4	The IIPARM panel. This enables you to enter additional parameters for the IIPARM command procedure based on your system requirements. This command procedure must be executed before any gateway utility session can be invoked.

Field	Description
Option 5	The main menu for Net.
Option 6	The main menu for invoking DBMS utilities.
Option 7	The main menu for creating JCL to compile gateway ESQL applications in WhiteSmiths C and PL/I.

Option 0: Gateway Facility Profile Panel

The Gateway Facility Profile Panel allows you to predefine a series of variables that are used by the Gateway Facility DBA utilities.

To invoke this panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section of this chapter.
2. Enter 0 in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

----- EDBC/Gateway Facility Profile Panel -----
IIPARM => ISUR(EDBC) PREFIX(EDBC.USAMGW) SABE(DEMO)
INGAREA => EDBC.USAMGW.INGAREA.CAIDEM1
INGJCL1 => //IGWFJOBS JOB (CAIDEM1),NOTIFY=CAIDEM1.CLASS=A
INGJCL2 => // MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=CAIDEM1,REGION=6M
INGJCL3 => //***
INGJCL4 => //***
INGPREFIX => EDBC.USAMGW
IIPREFIX =>
INGMODGN =>

INGISUR => ISUREDBC   INGSSABE => SABEDEMO
INGDRIVR => LOCAL     EDCUSER  => CAIDEM1   INGAUTH => N
INGULS   => MUSUOL   INGSYSDA => UIO       INGUNT  => 3380

IIIMS   =>           IIMSID   =>
IIUPDATE => -U       IIMAXC  => 999
INGARPRI => 2700    INGARSEC => 540
INGSRPRI => 150    INGSRSEC => 150
INGDB2IS =>        INGDB2AD =>

F1=HELP   F2=SPLIT  F3=END    F4=RETURN  F5=RFIND  F6=RCHANGE
F7=UP     F8=DOWN   F9=SWAP   F10=LEFT   F11=RIGHT F12=RETRIEVE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----
3/18 | CAPS | 14:12:50 | IBM-3278-2

```

Using This Panel

From this panel, you can perform either or both of the following operations:

- Overwrite the default values if necessary, and then press Enter to save the values and exit.

- Press PF3 to return to the IGWFDBA Main Menu.

Parameters on This Panel

All the default values for the parameters on this panel are set during the installation process. These parameters are:

IIPARM	=>	Contains the IIPARM Clist input parameters used in generating JCL or starting IIPARM command procedure. For example, for the checkout procedure of the VSAM gateway, type: ISVR (server logical symbols member) PREFIX (gateway data set prefix) SABE (DEMO) For more information on the IIPARM Clist, see the "Gateway Logical Symbols and IIPARM Clist" appendix.
INGAREA	=>	Gateway area data set name. For the gateway verification procedure, type the gateway data set prefix, followed by .INGAREA.DEMO .
INGJCL1-	=>	Contain JCL job control card statements INGJCL4 that set defaults used by all the skeleton options IGWFDBA invokes.
INGPREFIX	=>	Prefix for all gateway data sets.
IIPREFIX	=>	Prefix for IMS data set. Leave blank if not applicable.
INGMODGN	=>	Name of MODGEN library.
INGSISVR	=>	Logical symbols member used to configure the EDBC server. Type ISVR , followed by the name assigned to this member in the IGWFARM macro.
INGSSABE	=>	Logical symbols member used to access the specified user's area and sort work data set. For the gateway verification procedure, type SABE DEMO .
INGDRIVR	=>	Logical symbols member used to configure the gateway driver for this partition. For VSAM gateway verification, type LOCAL . Leave blank if not applicable.

EDCUSER	=>	User ID for the gateway user whose profile you are creating. For gateway verification, type your user ID as gateway DBA.
INGAUTH	=>	Indicates whether the modules used by IGWFDDBA reside in an APF-authorized library. Valid values are Y (Yes) and N (No). If the gateway modules are not within an APF-authorized library, you must specify N. If you do not, the IGWFDDBA facility fails with a 0C4 abend.
INGVLS	=>	VOLSER for all gateway data sets to be generated by IGWFDDBA.
INGSYSDA	=>	VOLSER for all temporary data sets to be generated by IGWFDDBA.
INGUNT	=>	UNIT value for all gateway data sets to be generated by IGWFDDBA.
IIMS	=>	Specifies the release of IMS to use with the gateway. Leave blank if not applicable.
IIMSID	=>	IMS subsystem name value, for invoking the gateway utilities with the IMS gateway. The default parameter is IMSI (for the IMS gateway). Leave blank if not applicable.
IIUPDATE	=>	Partition update intent flag. For update intent, enter -U. Leave blank if not applicable.
IIMAXC	=>	Partition maximum number of calls flag. For example, type 999.
INGARPRI	=>	The primary page allocation of the gateway area data set.
INGARSEC	=>	The secondary page allocation of the gateway area data set.
INGSRPRI	=>	The primary page allocation of the gateway sort data set.
INGSRSEC	=>	The secondary page allocation of the gateway sort data set.
INGDB2IS	=>	DB2 isolation level. Leave blank if not applicable.
INGDB2AD	=>	DB2 plan name for DSNTIAD. Leave blank if not applicable.

Option 2: VSAM Gateway Facility Main Menu

The VSAM Gateway Facility Main Menu provides a series of options to maintain and access the VSAM gateway.

To use this menu:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter **2** in the Selection field and press Enter.

The terminal displays a menu similar to the following:

```

----- EDBC/VSAM Gateway Facility Main Menu -----
SELECTION ==> 2                                USERID - MURTO01
                                                TIME   - 18.04

 1 ) (Reserved option for future usage)
 2 ) Display Active Gateway Drivers
 3 ) (Reserved option for future usage)
 4 ) Access EDBC Gateway Using File Input
 5 ) Access EDBC Gateway Using Terminal Monitor
 6 ) Create / Submit JCL to INVOKE EDBC DI Utilities
 7 ) Create / Submit JCL to FORMAT an EDBC area
 8 ) Create / Submit JCL to CREATE a user partition

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER your selection )

F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=RFIND   F6=RCHANGE
F7=UP        F8=DOWN      F9=SWAP     F10=LEFT   F11=RIGHT  F12=RETRIEVE
ATTN Clear Erase EOF Print Scrn Home Enter New Line Tab PA1 PA2 PA3 Reset
2/18 CAPS 15:08:29 IBM-3278-2

```

3. From this panel, enter the number of your choice in the Selection field and press Enter.

The terminal displays the panel for the option you request.

Option 2.2: Display Active Gateway Drivers Panel

The Gateway Display Active Drivers panel displays all currently active drivers of the gateway.

Using This Panel

To use this panel:

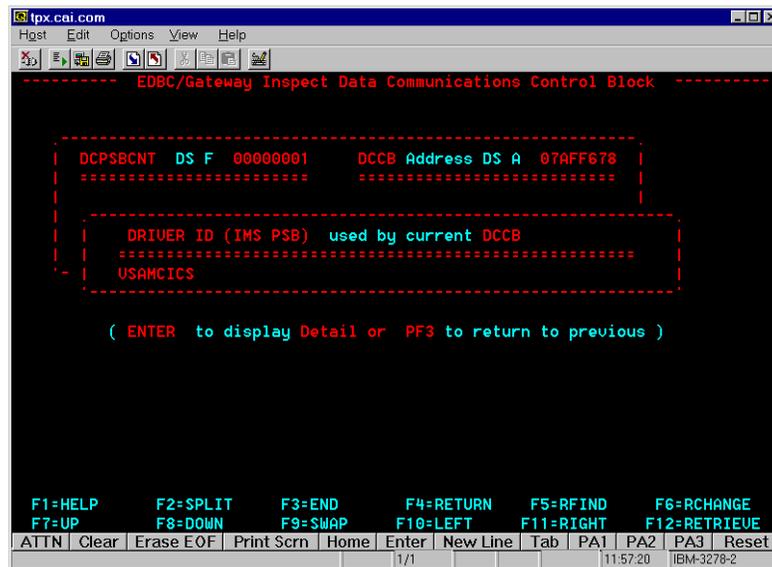
Field	Description
USER AUTH ID	The authorization ID of gateway user.
DRIVER ASID	The driver address space ID.
DRIVER JOBNAME	The procedure or jobname of gateway driver.
COMMENTS	Driver status.

Inspecting a Driver's Data Communication Control Block

To view a driver's Data Communication Control Block:

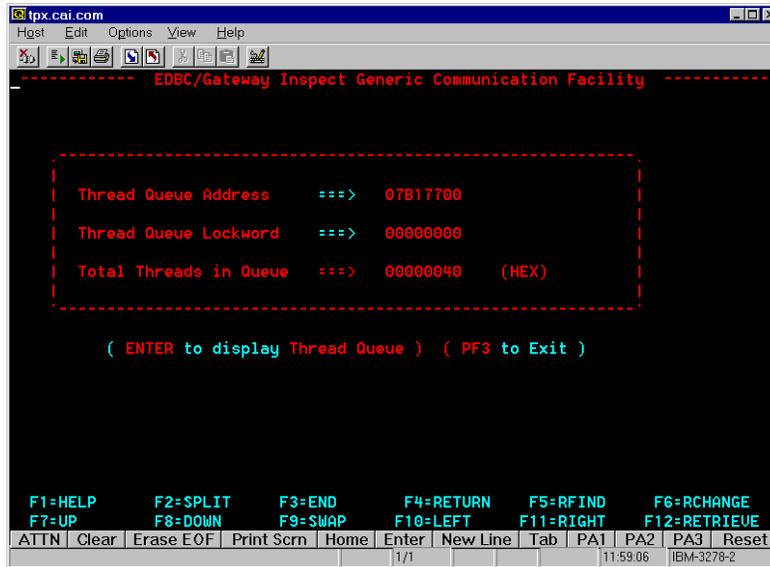
1. Type I in the CMD field of the Gateway Display Active Gateway Drivers panel.

The terminal displays a panel similar to the following:



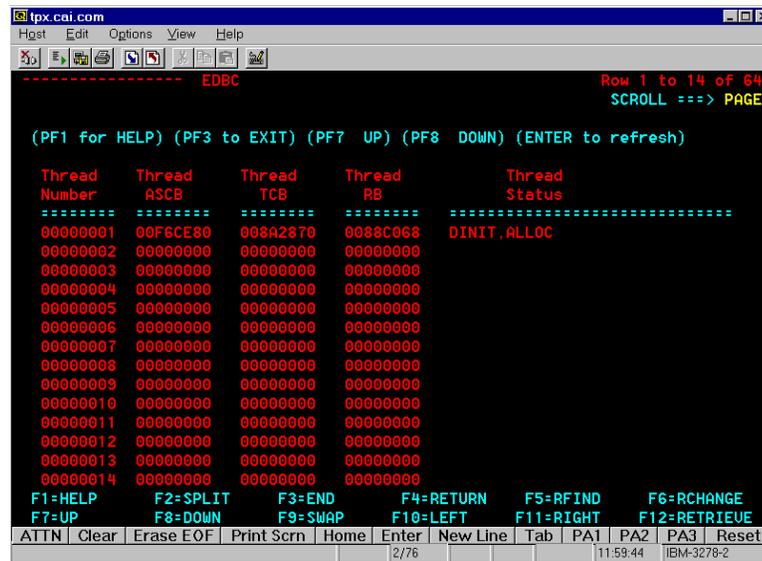
If you want to display the Data Communication Control Block information in greater detail, press Enter.

The terminal displays a panel similar to the following:



- If you want to display the status of each thread in the driver queue, press Enter.

The terminal displays a panel similar to the following:



You can press Enter to refresh this panel.

- Press PF3 to return to the calling panel.
2. Press PF3 to return to the Gateway Display Active Gateway Drivers panel.

Option 2.4: Gateway Using File Input Panel

The GUFU Gateway Using File Input panel allows you to invoke a gateway session as a batch program using batch JCL.

To invoke this panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2.4 in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

tpx.cai.com
Host Edit Options View Help
----- EDBC/GUFU Gateway Using File Input -----
USERID - MURTO01
TIME - 18:20
SYSTEM - XAE1

ENTER THE REQUIRED PARAMETERS:
Partition => IIUPUSM /*Partition Name */
Server => ISUREDBC /*Logical symbol of EDBC Server */
Inarea => SABEDEMO /*Logical symbol of User area */
Prefix => EDBC.USAMGW /*EDBC files prefix */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP F2=SPLIT F3=END F4=RETURN F5=RFIND F6=RCHANGE
F7=UP F8=DOWN F9=SWAP F10=LEFT F11=RIGHT F12=RETRIEVE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
11/20 | CAPS | 15.24.46 | IBM-3278-2

```

Using This Panel

From this panel, you can perform either or both of the following operations:

- Start the gateway session as a batch program.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel shows the following parameters:

PARTITION => Name of the gateway partition the gateway will access.

SERVER	=>	Name of server logical symbol member in <i>prefix.FILES.IIPARM</i> . For example: ISVREDBC
INGAREA	=>	The logical symbol member name of the gateway area. For example: SABEDEM0
PREFIX	=>	The prefix of gateway files. For example: EDBC.V2R3

Starting the Gateway Session as a Batch Program

To start the gateway session as a batch program:

1. Enter and verify the parameters shown on the panel and press Enter.
The terminal displays the temporary JCL.
2. Verify the temporary JCL.
If necessary, edit the temporary JCL. Verify that the batch input file in the SYSIN DD statement is correct and change it if necessary.
3. Submit the job by typing submit at the command line and pressing Enter, or press PF3 to exit.

Option 2.5: Gateway Using Terminal Monitor Panel

The GUTM Gateway Using Terminal Monitor panel allows you to use the Terminal Monitor to access the gateway DBA's partition and to issue sample relational queries against the sample installation databases. The Terminal Monitor runs under TSO.

To invoke this panel, use one of the following methods.

Invoking the panel directly:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section.
2. From the IGWFDDBA Main Menu, enter 2.5 in the Selection field and press Enter.

Invoking the panel sequentially:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section.

2. From the IGWFDDBA Main Menu, enter 2 in the Selection field and press Enter.
3. From VSAM Gateway Main Menu, enter 5 in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

tpx.cai.com
Hggt Edit Options View Help
----- EDBC/GUTM Gateway Using Terminal Monitor -----
COMMAND ==>
USERID - MURTO01
TIME - 14:06
SYSTEM - XAE1

==> SQL iipusm/vsam

----- IIPARM profile parameters -----
INGPREFIX ==> EDBC.USAMGW
INGSSABE ==> SABEDEMO

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to process )

F1=HELP F2=SPLIT F3=END F4=RETURN F5=RFIND F6=RCHANGE
F7=UP F8=DOWN F9=SWAP F10=LEFT F11=RIGHT F12=RETRIEUE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
5/11 | CAPS | 11.10.09 | IBM-3278-2

```

Using This Panel

From this panel, you can perform either or both of the following operations:

- Start the Terminal Monitor.
- Press PF3 to exit.

Parameters on This Panel

The IIPARM parameters shown on this panel are:

INGPREFIX	=>	The prefix for all gateway data sets.
INGSSABE	=>	Logical symbols member used to access the specified user's area and sort work data set.
INGSSABE	=>	Logical symbols member used to configure the EDBC server.

Invoking the Terminal Monitor

To start a Terminal Monitor session:

1. Verify the IIPARM profile parameters and change them if necessary.
2. Enter the name of the partition and the server class in the SQL Selection line.
For example, type:
iivpvsm/vsam
3. Press Enter to start the Terminal Monitor.
You can issue SQL queries against the sample databases provided on the installation tape. For more information on Terminal Monitor commands, see the *EDBC OpenSQL Reference Guide*.
4. Type `\q` and press Enter to exit from the Terminal Monitor.
5. Press PF3 to return to the calling panel.

Option 2.6: Create JCL to Invoke DI Utilities Panel

The Gateway Create JCL to Invoke DI Utilities panel allows you to select a DI utility to perform administration and management of the gateway area.

To invoke this panel, use one of the following methods.

Invoke the panel directly:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2.6 in the Selection field and press Enter.

Invoke the panel sequentially:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2 in the Selection field and press Enter.
3. From VSAM Gateway Main Menu, enter 6 in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

tpx.cai.com
-----
SELECTION ===>
                                USERID - MURTO01
                                TIME    - 14:59
                                SYSTEM  - XAE1

Create JCL to INVOKE DI Utilities:

1 ) DIATEDMP  /* Show space allocation in an INGAREA          */
2 ) DIDIRDMP  /* Directory information of File,Partition,INGAREA*/
3 ) DIBACKUP  /* Backup an INGAREA with options                          */
4 ) DIRESTOR  /* Restore an INGAREA or Scan Backup file                  */
5 ) DICOMPRS  /* Backup and Restore an INGAREA with Compress             */
6 ) DICOPYAR  /* Copy a Source Area to a Target Area                     */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER your selection )

F1=HELP      F2=SPLIT      F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
F7=UP        F8=DOWN        F9=SWAP     F10=LEFT    F11=RIGHT    F12=RETRIEU
ATN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----
2/18      CAPS      12.03.15  IBM-3278-2

```

Using This Panel

From this panel, you can perform either or both of the following operations:

- Type a selection number in the Selection field and press Enter. The terminal displays the panel that corresponds to your entry.
- Press PF3 to return to the calling panel.

Option 2.6.1: Gateway Facility to Create JCL to Run diatedmp Utility Panel

This panel allows you to build and submit JCL to run the diatedmp utility, which shows the number of 2 KB pages contained in the gateway area and the number of pages allocated and used by each partition.

To invoke the Gateway Create JCL to Run DIATEDMP Utility panel, use one of the following methods.

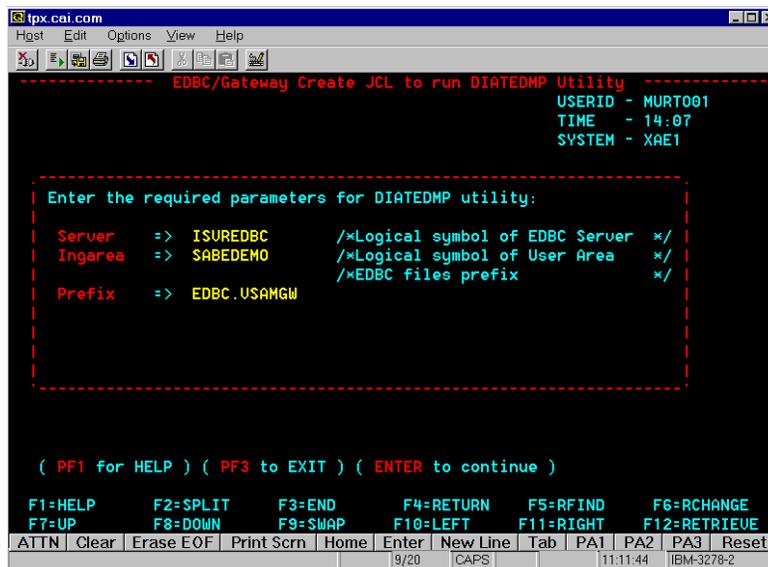
Invoke the panel directly:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2.6.1 in the Selection field and press Enter.

Invoke the panel sequentially:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2 in the Selection field and press Enter.
3. From VSAM Gateway Main Menu, enter 6 in the Selection field and press Enter.
4. From the DI Utility Main Menu, enter 1 in the Selection field and press Enter.

The terminal displays a panel similar to the following:



Using This Panel

From the Gateway Facility to Create JCL to Run DIATEDMP Utility panel, you can perform either or both of the following operations:

- Enter the parameters to run the diatedmp utility.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel displays the following parameters:

SERVER => The name of the server logical symbols member in *prefix.FILES.IIPARM*.
 For example: ISVREDBC

INGAREA	=>	The logical symbol member that associates a gateway area and sort work data set.
PREFIX	=>	The prefix for all gateway data sets. For example: EDBC.V2R3

Running the
diatedmp Utility

To run the diatedmp utility:

1. Enter the parameters and press Enter.
The temporary JCL appears.
2. Review and customize the JCL if necessary.
3. Type **submit** in the command line to submit the JCL and run the diatedmp utility.

Option 2.6.2: Gateway Facility to Create JCL to Run didirdmp Utility Panel

This panel allows you to build and submit JCL to run the didirdmp utility, which allows the dump of the directory associated with a specific partition. You can use this option to show the DI-files associated with a partition and the associated control blocks.

Specifying a file name in the File Name parameter will invoke the DI utility to dump the content of the specified file.

To invoke the Gateway Facility to Create JCL to Run DIDIRDMP Utility panel, use one of the following methods.

Invoke the panel directly:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section.
2. From the IGWFDDBA Main Menu, enter 2.6.2 in the Selection field and press Enter.

Invoke the panel sequentially:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section.
2. From the IGWFDDBA Main Menu, enter 2 in the Selection field and press Enter.
3. From VSAM Gateway Main Menu, enter 6 in the Selection field and press Enter.

- From the DI Utility Main Menu, enter 2 in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

----- EDBC/Gateway Create JCL to run DIDIRDMP Utility -----
                                USERID - WURT001
                                TIME   - 14:08
                                SYSTEM - XAE1

Enter the required parameters for DIDIRDMP utility:

Server  => ISUREDBC      /*Logical symbol of EDBC Server */
Ingarea => SABEDMO      /*Logical symbol of User Area  */
Prefix  => EDBC.USAMGW  /*EDBC files prefix           */

Partition => IIUPUSM    /*Partition Name               */
File Name =>            /*Specific DI-file name to dump*/

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=RFINDD   F6=RCHANGE
F7=UP        F8=DOWN     F9=SWAP    F10=LEFT   F11=RIGHT  F12=RETRIEU
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
          |9/20| CAPS |          |11.12.51| IBM-3279-2
    
```

Using This Panel

From the Gateway Facility to Create JCL to Run DIDIRDMP Utility panel, you can perform either or both of the following operations:

- Enter the parameters to run the didirdmp utility.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel shows the following parameters:

- | | | |
|---------|----|---|
| SERVER | => | The name of the server logical symbols member in <i>prefix.FILES.IIPARM</i> .
For example: ISVREDBC |
| INGAREA | => | The logical symbol member that associates a gateway area and sort work data set. This logical symbol member is stored in <i>prefix.FILES.IIPARM</i> .
For example: SABEDMO |
| PREFIX | => | The prefix for all gateway data sets.
For example: EDBC.V2R3 |

- PARTITION => The partition name that is accessed by the didirdmp utility.
For example: IIVPVSM
- FILE NAME => Enter the DI-file name that is passed to the didirdmp utility to dump its contents.
You can leave this field blank.

Running the didirdmp Utility

To run the didirdmp utility:

1. Enter the parameters and press Enter.
The temporary JCL appears.
2. Review and customize the JCL as required.
3. Type **submit** in the command line to submit the JCL and run the didirdmp utility.

Option 2.6.3: Gateway Facility to Create JCL to Run dibackup Utility Panel

This panel allows you to build and submit JCL to run the dibackup utility. This utility allows you to specify and back up a gateway area. The backup file is created in a permanent data set. You can use the Options field to pass additional options to this utility.

To invoke the Gateway Facility to Create JCL to Run DIBACKUP Utility panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2.6.3 in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

tpx.cai.com
----- EDBC/Gateway Create JCL to run DIBACKUP Utility -----
                                USERID - MURTO01
                                TIME   - 14:09
                                SVSTEM - XAE1

Enter the required parameters for DIBACKUP utility:

Server  => ISVREDBC      /*Logical symbol of EDBC Server */
Ingarea => SABEDMO      /*Logical symbol of User Area  */
Prefix  => EDBC.USAMGW  /*EDBC files prefix           */
Backup  => EDBC.USAMGW.INGAREA.DEMO.BACKUP /*EDBC Area Backup dataset name*/
Options =>              /*Enter R or S or C or Blank  */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP   F2=SPLIT   F3=END   F4=RETURN  F5=RFIND  F6=RCHANGE
F7=UP     F8=DOWN   F9=SWAP  F10=LEFT  F11=RIGHT F12=RETRIEUE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
9/20  CAPS  11:13:45  IBM-3278-2
    
```

Using This Panel

From the Gateway Facility to Create JCL to Run DIBACKUP Utility panel, you can perform either or both of the following operations:

- Enter the parameters to run the dibackup utility.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel shows the following parameters:

- | | | |
|---------|----|---|
| SERVER | => | The name of the server logical symbols member in <i>prefix.FILES.IIPARM</i> .

For example: ISVREDBC |
| INGAREA | => | The logical symbol member that associates a gateway area and sort work data set. This logical symbol member is stored in <i>prefix.FILES.IIPARM</i> .

For example: SABEDMO |
| PREFIX | => | The prefix for all gateway data sets.

For example: EDBC.V2R3 |
| BACKUP | => | The data set name of the backup file.

For example: EDBC.V2R3.INGAREA.DEMO.BACKUP |

- OPTIONS => Additional options for thedibbackup utility. Valid entries are:
- ' ' - Specifies no parameter. Generates minimal dibbackup messages.
 - C - COMPRESS option. This option generates the same style report as the SQL_REPORT option, but does not write any files that are eligible for purge to the backup file.
 - R - REPORT option. This option generates a detailed report showing the operation of this utility.
 - S - SQL_REPORT option. This option generates a SQL INSERT script that inserts value into tables created by the IIDICING SQL script. You can use the IIDIQDET or IIDIQSUM SQL scripts to produce a report about the gateway area.

Running the
dibbackup Utility

To run the dibbackup utility:

1. Enter the parameters and press Enter.
The temporary JCL appears.
2. Review and customize the JCL as required.
3. Type **submit** in the command line to submit the JCL and run the dibbackup utility.

Option 2.6.4: Gateway Facility to Create JCL to Run direstor Utility Panel

This panel allows you to build and submit JCL to run the direstor utility. This utility allows you to restore a backup file to the specified gateway area. It also provides an option to scan a BACKUP file and produce a report on it.

To invoke the Gateway Facility to Create JCL to Run DIRESTOR Utility panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2.6.4 in the Selection field and press Enter.

INGAREA	=>	The logical symbol member that associates a gateway area and sort work data set. This logical symbol member is stored in <i>prefix.FILES.IIPARM</i> . For example: SABEDEM0
PREFIX	=>	The prefix for all gateway data sets. For example: EDBC.V2R3
BACKUP	=>	The data set name of the BACKUP file. For example: EDBC.V2R3.INGAREA.DEMO.BACKUP
TYPE	=>	Processing types for the direstor utility. Valid entries are: <ul style="list-style-type: none">■ R - Restore option. This option restores the BACKUP file to the specified gateway area dsname.■ S - Scan option. This option scans the BACKUP file and produces a report containing the DI-files contained on that backup.
INGAREA DSNAME	=>	The data set name of the gateway area for the restore process. For example: EDBC.V2R3.INGAREA.DEMO

Option 2.6.5: Gateway Facility to Create JCL to Run dicomprs Utility Panel

This panel allows you to build and submit JCL to run the dicomprs utility, which consists of invocations of dibackup with the COMPRESS option and direstor. The effect is to delete any unused DI-files in a gateway area, thus freeing space to be used by the tables in the gateway area. The BACKUP file that is created by this utility is a temporary file and will be deleted by the end of the process.

To invoke the Gateway Facility to Create JCL to Run DICOMPRS Utility panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2.6.5 in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

----- EDBC/Gateway Create JCL to run DICOMPRS Utility -----
                                USERID - MURTO01
                                TIME   - 14:12

Enter the required parameters for DICOMPRS utility:

Server    => ISUREDBC    /*Logical symbol of EDBC Server */
Ingarea   => SABEDEM0    /*Logical symbol of User Area   */
Prefix    => EDBC.USAMGW /*EDBC files prefix             */
Ingarea DSname => EDBC.USAMGW.INGAREA.DEMO /*EDBC Area dataset name       */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP    F2=SPLIT    F3=END    F4=RETURN    F5=RFIND    F6=RCHANGE
F7=UP      F8=DOWN     F9=SWAP   F10=LEFT    F11=RIGHT   F12=RETRIEVE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset |
      8/22  CAPS      11:16:15  IBM-3278-2
    
```

Using This Panel

From the Gateway Facility to Create JCL to Run DICOMPRS Utility panel, you can perform either or both of the following operations:

- Enter the parameters to run the dicomprs utility.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel displays following parameters:

- | | | |
|-------------------|----|---|
| SERVER | => | The name of the server logical symbols member in <i>prefix.FILES.IIPARM</i> .
For example: ISVREDBC |
| INGAREA | => | The logical symbol member that associates a gateway area and sort work data set. This logical symbols member is stored in <i>prefix.FILES.IIPARM</i> .
For example: SABEDEM0 |
| PREFIX | => | The prefix for all gateway data sets.
For example: EDBC.V2R3 |
| INGAREA
DSNAME | => | The data set name of the gateway area for the restore process.
For example: EDBC.V2R3.INGAREA.DEMO |

Running the dicomprs
Utility

To run the dicomprs utility:

1. Enter the parameters and press Enter.
The temporary JCL appears.
2. Review and customize the JCL as required.
3. Type **submit** in the command line to submit the JCL and run the dicomprs utility.

Option 2.6.6: Gateway Facility to Create JCL to Run dicopyar Utility Panel

This panel allows you to build and submit JCL to run the dicopyar utility, which consists of invocations of dibackup with the COMPRESS option and direstor. You can use this utility to create the local environment for accessing a new gateway area for a new user. You can also use dicopyar to copy an existing area into a new gateway area.

To invoke the Gateway Create JCL to Run DICOPYAR Utility panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2.6.6 in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

tpx.cai.com
-----
EDBC/Gateway Create JCL to run DICOPYAR Utility
USERID - MURTO01
TIME   - 14:13

Enter the required parameters for DICOPYAR utility:

Server   => ISUREDBC   /*Logical symbol of EDBC Server */
Prefix   => EDBC.USAMGW /*EDBC files prefix */

Source Area => SABEDEMO /*Logical symbol of Source Area */
Target Area => UCAIDEM1 /*Logical symbol of Target Area */
Target User => CAIDEM1 /*Userid for Target User */
Target DSname => EDBC.USAMGW.INGAREEA.CAIDEM1 /*EDBC Area dataset name */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )
F1=HELP      F2=SPLIT      F3=END        F4=RETURN     F5=RFIND      F6=RCHANGE
F7=UP        F8=DOWN       F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----
8/22 CAPS 11:17:35 IBM-3278-2
    
```

Using This Panel

From the Gateway Facility to Create JCL to Run DICOPYAR Utility panel, you can perform either or both of the following operations:

- Enter the parameters to run the dicopyar utility.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel shows the following parameters:

SERVER	=>	The name of the server logical symbols member in <i>prefix.FILES.IIPARM</i> . For example: ISVREDBC
PREFIX	=>	The prefix for all gateway data sets. For example: EDBC.V2R3
SOURCE AREA	=>	The name of the logical symbol of the source area. For example: SABEDEMO
TARGET AREA	=>	The name of the logical symbol of the target area. For example: UCAIDEM1
TARGET USER	=>	The user ID for the target user. For example: CAIDEM1

TARGET The data set name of the target gateway area.
 DSNAME => For example: EDBC.V2R3.INGAREA.CAIDEM1

Running the Dicopyar Utility To run the dicopyar utility:

1. Enter the parameters and press Enter.
 The temporary JCL appears.
2. Review and customize the JCL if required.
3. Type **submit** in the command line to submit the JCL and run dicopyar.

Option 2.7: Create JCL to Format a Gateway Area Panel

This panel allows you to build and submit JCL to format a gateway area. This process is required before any user database can be created for the VSAM gateway.

To invoke the Gateway Create JCL to Format an EDBC Area panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 2.7 in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

tpx.cai.com
----- EDBC/Gateway Create JCL to FORMAT an EDBC Area -----
                                USERID - MURTO01
                                TIME    - 14:14

ENTER THE REQUIRED PARAMETERS:
| Server  => ISUREDBC           /*Logical symbol of EDBC Server */
| Ingarea => SABEDEMO           /*Logical symbol of User Area  */
| Userid  => CAIDEM1           /*EDBC logical userid         */
|                               /*EDBC Area dataset name      */
| Ingarea DSName => EDBC.USAMGW.INGAREA.CAIDEM1

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP   F2=SPLIT   F3=END   F4=RETURN   F5=RFIND   F6=RCHANGE
F7=UP     F8=DOWN    F9=SWAP  F10=LEFT  F11=RIGHT  F12=RETRIEU
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
10/19 | CAPS | 11:18:29 | IBM-3278-2
  
```

Using This Panel

From the Gateway Create JCL to Format an EDBC Area panel, you can perform either or both of the following operations:

- Enter the parameters to format a gateway area.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel shows the following parameters:

SERVER	=>	The name of the server logical symbol member in <i>prefix.FILES.IIPARM</i> . For example: ISVREDBC
INGAREA	=>	The logical symbol member that associates the given user ID with a given area and sort work data set. This logical symbol member is stored in <i>prefix.FILES.IIPARM</i> .
USERID	=>	The user ID to whom the area is assigned.
INGAREA DSNAME	=>	The name of the area data set you are formatting. Enter the fully qualified name of the data set.

Formatting a Gateway Area

To format a gateway area:

1. Enter the parameters and press Enter.
The temporary JCL appears.
2. Review and customize the JCL as required.
3. Type **submit** in the command line to submit the JCL and allocate and format a gateway area.

Option 2.8: Create JCL to Create a User Partition Panel

The VSAM Create JCL to Create a User Partition panel allows you to build and submit JCL to create a gateway user partition.

To invoke this panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.

- From the IGWFDDBA Main Menu, enter **2.8** in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

tpx.cai.com
-----
EDBC/USAM Create JCL to CREATE a User Partition
USERID - MURTO01
TIME   - 14:15

ENTER THE REQUIRED PARAMETERS:

Partition => IIUPUSM /*Partition for USAM Database */
Server    => ISUREDBC /*Logical symbol of EDBC Server */
Ingarea   => SABEDEM0 /*Logical symbol of User area */
Usam Driver => LOCAL /*Specify USAM driver id */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP F2=SPLIT F3=END F4=RETURN F5=RFIND F6=RCHANGE
F7=UP F8=DOWN F9=SWAP F10=LEFT F11=RIGHT F12=RETRIEVE
ATTN Clear Erase EOF Print Scrn Home Enter New Line Tab PA1 PA2 PA3 Reset
10/22 CAPS 11.19.21 IBM-3278-2

```

Using This Panel

From this panel, you can perform either or both of the following operations:

- Create a user partition.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel shows the following parameters:

PARTITION	=>	User partition name for the VSAM Database.
SERVER	=>	Name of server logical symbols member in the <i>prefix.FILES.IIPARM</i> . For example: ISVREDBC
INGAREA	=>	The logical symbol member for the user in question. This logical symbol member is stored in <i>prefix.FILES.IIPARM</i> .
VSAM DRIVER	=>	CICS VSAM driver name or LOCAL (if VSAM data set is not under CICS control). For example: LOCAL

Using This Panel

From this panel, you can perform any or all of the following operations:

- Invoke the IIPARM command procedures.
- Enter help on the IIPARM parameter line and press Enter to invoke the help command procedure.
- Press PF3 to return to the IGWFDBA Main Menu.

Parameters on This Panel

This panel shows the following parameters:

IIPARM Parameters from Profile	=>	Displays the default IIPARM parameter that is set in the profile panel. You can modify the default IIPARM parameter by typing over the default values. The default parameter is: ISVR(EDBC) PREFIX(EDBC.V2R3) SABE(DEMO)
USER	=>	Specifies the database area for a user.
DSN	=>	Specifies the IIPARM partition data set name.
DBNM	=>	Specifies database access logical member for remote connection.
ERLOG	=>	Specifies IIERLOG allocation.
STOGRP	=>	Allocates IISTOGRP. The valid entries are: <ul style="list-style-type: none"> ■ NONE – Specifies that the IISTOGRP is not allocated. ■ Dummy – Specifies that the IISTOGRP is allocated as a dummy data set. ■ YES – Specifies that the IISTOGRP is allocated; uses the PREFIX parameter.
STOLOG	=>	Specifies dsname of the DD statement of IISTOLOG.
AREA	=>	Specifies the database area for a user.

Invoking the IIPARM Command Procedures

To invoke the IIPARM command procedures, enter the IIPARM parameters and press Enter. Upon successful completion of the command procedure, the terminal displays a completion message.

Option 5: Net Utilities Menu Panel

The Net Utilities Menu panel is the main menu for the netu and iinamu utilities.

To invoke this panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter **5** in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

tpx.cai.com
----- EDBC/NET Utilities Menu -----
SELECTION ==>                                USERID - MURTO01
                                              TIME   - 14:18
                                              SYSTEM - XAE1

  Invoke dynamically, the following INET Utilities.

      1) NETU
      2) IINAMU

  -- IIPARM Profile Parameters for INET Utilities --
  INGPRFIX ==> EDBC.USAMGW
  INGSISUR ==> ISUREDBC

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER your selection )

F1=HELP   F2=SPLIT   F3=END   F4=RETURN  F5=RFIND  F6=RCHANGE
F7=UP     F8=DOWN    F9=SWAP  F10=LEFT  F11=RIGHT F12=RETRIEUE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
                2/18 | CAPS |                11:22:31 | IBM-3278-2
  
```

Using This Panel

From this panel, you can perform one or more of the following operations:

- Invoke the netu utility.
- Invoke the iinamu utility.
- Press PF3 to return to the IGWFDBA Main Menu.

Fields for this Panel

This panel shows the following fields:

Field	Description
INGPRFIX	The prefix for all gateway data sets.
INGSISVR	The logical symbols member used to configure the EDBC server.

Invoking the Netu Utility

To invoke the netu utility:

1. Review the IIPARM profile parameters and change them as necessary.
2. Enter **1** in the Selection line and press Enter.

After the process completes, the panel appears again and you can make another selection.

For more information on the netu utility, see the *Net User Guide*.

Invoking the iinamu
Utility

To invoke the iinamu utility:

1. Review the IIPARM profile parameters and change them as necessary.
2. Enter **2** in the Selection line and press Enter.

When the processing is complete, the panel appears again and you can make another selection.

For more information on the iinamu utility, see the *Net User Guide*.

Option 6: DBMS Utility Menu

The Gateway DBMS Utility Menu allows you to invoke various DBMS utilities to manage gateway areas and partitions. You can invoke the following utilities from this menu:

```
createpr  
dbconst  
vwconst  
destroypr  
finddb  
verifydb
```

To invoke this menu:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter **6** in the Selection field and press Enter.

The terminal displays a panel similar to the following:

```

tpx.cai.com
----- EDBC/Gateway DBMS Utility Menu -----
SELECTION ==> _                               USERID - MURTO01
                                              TIME   - 18:44
                                              SYSTEM - XAE1

Invoke dynamically, the following EDBC Utilities.

1) CREATEPR
2) DBCONST
3) UWCONST
4) DESTROY
5) FINDDB
6) VERIFVDB

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER your selection )

F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
F7=UP        F8=DOWN     F9=SWAP    F10=LEFT   F11=RIGHT   F12=RETRIEU
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
2/18 | ICAPS | 15.47.46 | IBM-3278-2
    
```

Using This Panel

From this panel, you can perform either or both of the following operations:

- Enter the number of your option choice in the Selection line and press Enter.
- Press PF3 to return to the IGWFDBA Main Menu.

Using the DBMS Utilities

To use a DBMS utility:

1. Enter the number of your selection in the Selection line of the Gateway DBMS Utility Menu panel.

The terminal displays a panel similar to the following:

```

tpx.cai.com
----- EDBC/Gateway Create JCL to COMPILE application in WhiteSmith C -----
USERID - MURTO01
TIME   - 14:21

ENTER THE REQUIRED JCL SYMBOLICS:

EDBC Dataset Prefix => EDBC.USAMGW
Server              => ISUREDBC

Source Library      => EDBC.USAMGW.SAMPLE.C
Source Member       => EDCAPP1
Load Library        => EDBC.USAMGW.USER.LOAD
Load Member         => EDCAPP1

Additional Libraries? (Y/N) => Y
ESQL Application?   (Y/N) => Y

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP   F2=SPLIT   F3=END     F4=RETURN  F5=RFIND   F6=RCHANGE
F7=UP     F8=DOWN    F9=SWAP   F10=LEFT  F11=RIGHT  F12=RETRIEUE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
8/30 | ICAPS | 11.26.10 | IBM-3278-2

```

Using This Panel

From this panel you can perform either or both of the following operations:

- Enter the parameters to create the temporary JCL.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel shows the following list of parameters:

EDBC Data Set Prefix	=>	The prefix for all gateway data sets. For example: EDBC.V2R3
Server	=>	The name of the server logical symbols member in <i>prefix.FILES.IIPARM</i> . For example: ISVREDBC
Source Library	=>	The OS/390 PDS library name that contains the source codes of the program.
Source Member	=>	The OS/390 PDS member name of the source codes.
Load Library	=>	The OS/390 PDS library name that stores the load modules.
Load Member	=>	The load module name of the source program.

- Additional Libraries? => Valid entries are:
- Y – Invokes another panel to enter additional libraries that are required to compile and link an application.
 - N – Indicates an additional library is not required.
- ESQL Application? => Valid entries are:
- Y – Indicates the source program is an ESQL application.
 - N – Indicates the source program is not an ESQL application.

Creating the JCL with Additional Libraries

To create the compilation JCL with additional libraries:

1. Enter the correct values for the parameters listed on the Panel. Enter **Y** in the Additional Libraries field.
2. When the entries are complete, press Enter.

The terminal displays a panel similar to the following:

```

----- EDBC/Gateway Add Additional Library to compile Procedure -----
                                USERID - MURTO01
                                TIME   - 19:00

| Enter additional STEPLIB library name(s):
| 1 Library ==>
| 2 Library ==>
| 3 Library ==>
|
| Enter additional SYSLIB library name(s):
| 4 Library ==>
| 5 Library ==>
| 6 Library ==>
|
| Enter additional INCLUDE library name(s):
| 7 Library ==>
| 8 Library ==>
|
| Enter SVSMSGS library name(s) for IBM C Compile:
| 9 Library ==>
|
| ( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )
F1=HELP   F2=SPLIT  F3=END    F4=RETURN  F5=RFIND  F6=RCHANGE
F7=UP     F8=DOWN   F9=SWAP   F10=LEFT  F11=RIGHT F12=RETRIEU
ATTN| Clear| Erase EOF| Print Scrn| Home| Enter| New Line| Tab| PA1| PA2| PA3| Reset
-----
                                6/22  CAPS  16:04:15  IBM-3278-2
    
```

3. Enter any additional libraries that are specific to your environment. The libraries you enter will be placed in the appropriate job steps in the temporary batch JCL. Enter the library name(s) without quotes.

4. When the entries are complete, press Enter.
The terminal displays a batch jobstream. The batch jobstream is temporary JCL and is deleted when you exit the panel.
5. Review and customize the JCL if necessary.
6. Type **submit** in the command line to submit the JCL to compile the program.
7. Press PF3 to return to the calling panel.

Creating the JCL Without Additional Libraries

To create the compilation JCL without additional libraries:

1. Enter the correct values for the parameters listed on the panel. Enter N in the Additional Libraries field.
2. When the entries are complete, press Enter.
The terminal displays a batch jobstream. The batch jobstream is temporary JCL and is deleted when you exit the panel.
3. Review and customize the JCL if necessary.
4. Type **submit** in the command line to submit the JCL to compile the program.
5. Press PF3 to return to the calling panel.

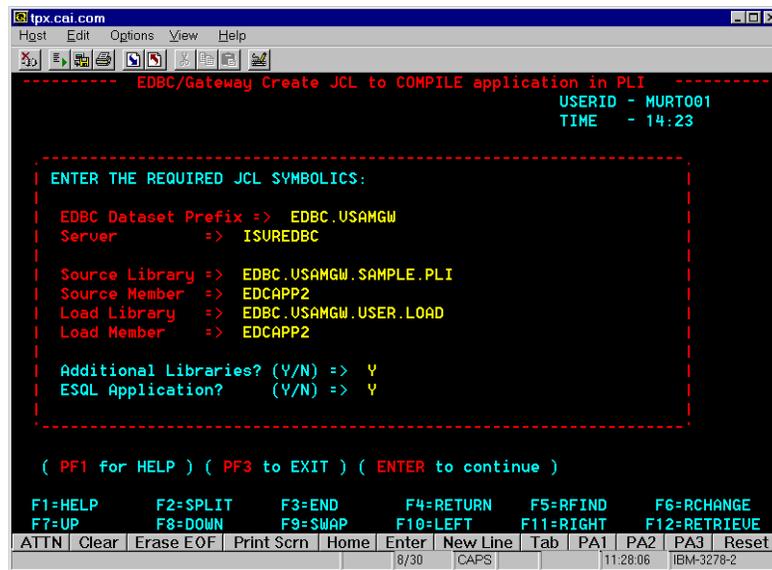
Option 7.2: Create JCL to Compile Application in PL/I Panel

This panel allows you to build and submit JCL to compile applications in PL/I.

To invoke the Gateway Create JCL to Compile Application in PL/I panel:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section.
2. From the IGWFDBA Main Menu, enter 7.2 in the Selection field and press Enter.

The terminal displays a panel similar to the following:



Using This Panel

From the Gateway Create JCL to Compile Application in PL/I panel, you can perform either or both of the following operations:

- Enter the parameters to create the temporary JCL.
- Press PF3 to return to the calling panel.

Parameters on This Panel

This panel shows the following list of parameters:

EDBC Data Set Prefix	=>	The prefix for all gateway data sets. For example: EDBC.V2R3
Server	=>	The name of the server logical symbols member in <i>prefix.FILES.IIPARM</i> . For example: ISVREDBC
Source Library	=>	The OS/390 PDS library name that contains the source codes of the program.
Source Member	=>	The OS/390 PDS member name of the source codes.
Load Library	=>	The OS/390 PDS library name that stores the load modules.
Load Member	=>	The load module name of the source program.

- Additional Libraries? => Valid entries are:
- Y – Invokes another panel to enter additional libraries that are required to compile and link an application.
 - N – Indicates an additional library is not required.
- ESQL Application? => Valid entries are:
- Y – Indicates the source program is an ESQL application.
 - N – Indicates the source program is not an ESQL application.

Creating the JCL with Additional Libraries

To create the compilation JCL with additional libraries:

1. Enter the correct values for the parameters listed on the panel. Enter **Y** in the Additional Libraries field.
2. Press Enter.

The terminal displays a panel similar to the following:

```

----- EDBC/Gateway Add Additional Library to compile Procedure -----
                                USERID - MURTO01
                                TIME   - 19:06

| Enter additional STEPLIB library name(s):
| 1 Library ==>  SYS2.PLI.LOAD
| 2 Library ==>  SYS2.PLI.LINKLIB
| 3 Library ==>
|
| Enter additional SYSLIB library name(s):
| 4 Library ==>  SYS2.PLI.LOAD
| 5 Library ==>  SYS2.PLI.LINKLIB
| 6 Library ==>
|
| Enter additional INCLUDE library name(s):
| 7 Library ==>
| 8 Library ==>
|
| Enter SYSMSGs library name(s) for IBM C Compile:
| 9 Library ==>
|
| ( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )
F1=HELP   F2=SPLIT  F3=END    F4=RETURN  F5=RFIND  F6=RCHANGE
F7=UP     F8=DOWN   F9=SWAP   F10=LEFT  F11=RIGHT F12=RETRIEUE
-----
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----
6/22 | CAPS | 16:10:27 | IBM-3278-2

```

3. Enter any additional libraries that are specific to your environment. The libraries you enter will be placed in the appropriate job steps in the temporary batch JCL. Enter the library name(s) without quotes.
4. When the entries are complete, press Enter.

The terminal displays a batch jobstream. The batch jobstream is temporary JCL and is deleted when you exit the panel.

5. Review and customize the JCL if necessary.
6. Type **submit** in the command line to submit the JCL to compile the program.
7. Press PF3 to return to the calling panel.

Creating the JCL
Without Additional
Libraries

To create the compilation JCL without additional libraries:

1. Enter the correct values for the parameters listed on the panel. Enter **N** in the Additional Libraries field.
2. When the entries are complete, press Enter.
The terminal displays a batch jobstream. The batch jobstream is temporary JCL and is deleted when you exit the panel.
3. Review and customize the JCL if necessary.
4. Type **submit** in the command line to submit the JCL to compile the program.
5. Press PF3 to return to the calling panel.

Configuring the EDBC Server

This chapter describes the procedures to:

- Install and configure communication interfaces
- Enable and test security interfaces

This chapter also describes EDBC configuration options:

- Force Inactive Timeout
- Local Time Zone
- Alternate Translation Tables

Install and Configure Communication Interfaces

EDBC supports the SNA_LU0, SNA_LU62, IBM TCP/IP, SNS TCP/IP, KNET TCP/IP, and CCI network protocols. The EDBC network configuration is defined when EDBC is installed by specifications in the stage1 IGWFPSRV statement.

Modifications to the network do not require that EDBC be reinstalled. Additions or changes can be made by updating and submitting job ASMPSERV in the SAMPLE.CNTL library.

EDBC network configuration is maintained as a load module with a default name of IIPSERV. An installation can maintain multiple network configurations. The network configuration used by EDBC is controlled by logical symbol II_NET_LMOD. The default is:

```
II_NET_LMOD = IIPSERV;
```

SNA LU0 for OS/390

EDBC supports the IBM SNA LU0 protocol on OS/390. This can be used for OS/390-to-OS/390 access as well as access from other platforms to OS/390.

Requirements

The gateway is certified for IBM OS/390 SNA LU0 with the following configuration: ACF/VTAM version 3.1.1 or higher.

Installation and Configuration

To install SNA LU0 support for the EDBC server, code the following parameters on the IGWFPSVR statement:

```
TYPE=SNA_LU0  
ACB=  
NODENAME=  
DLOGMODE=  
MODETAB=  
ENABLE=
```

VTAM ACB and LOGMODE definitions for the SNA_LU0 logical unit are generated and placed in the SAMPLE.CNTL library as members RTIAPPL and RTIMODE. These definitions should be given to personnel responsible for VTAM. The SNA_LU0 interface cannot be started until the VTAMLST and VTAMLIB libraries have been updated with the generated ACB and LOGMODE definitions and the newly defined ACB activated.

Starting and Stopping the SNA_LU0 Interface

The following logical in the ISVREDBC member of EDBC.V2R3.FILES.IIPARM determines whether the SNA_LU0 network interface is to be automatically started during server initialization:

```
II_PROTOCOL_SNA_LU0 = YES;
```

The ENABLE= parameter of the stage1 IGWFPSVR statement determines the value of this logical.

The alternative to starting the interface during server initialization is to activate it dynamically by issuing the following Modify operator command:

```
F EDBC,ACT,PROT=SNA_LU0
```

To stop the interface, issue the following Modify operator command:

```
F EDBC,INACT,PROT=SNA_LU0
```

Note: The above command terminates active SNA_LU0 connections.

Connecting from a Remote Client

To connect to the EDBC server using the SNA_LU0 interface, a vnode entry must be created on the EDBC client. See the “Managing Network Communications” chapter in *EDBC Getting Started* for information on how to create vnode entries for SNA_LU0.

SNA LU0 Abend Codes and Messages

Various internal error conditions can cause the SNA LU0 interface to terminate abnormally (abend). When an abend occurs, the EDBC server continues functioning; however, the SNA LU0 interface stops functioning. The abend code and summary information generally appear on the OS/390 console and the server JES log. The active load module displayed in the summary information is IILU0PS. The following table lists the possible abend codes:

Abend	GTF Error Message	Program
1001	ERROR WHEN SETTING UP RPL	PS0PSIN
1002	NO MATCHING PCB ADDR FOUND	Session Init
1003	NO CONNECT PWQE FOUND FOR THIS PCB	
1004	OPNDST ACCEPT MACRO ERROR	
1005	CLSDST MACRO ERROR	
1006	INVALID USER DATA - SESSION REJECTED	
1001	ERROR WHEN SETTING UP RPL	PS0RCVA
1002	ERROR WHEN REISSUING RECEIVE ANY	Receive Any
1003	RECEIVED A NEGATIVE RESPONSE	
1004	NO PCB FOUND FOR RCVD MSG WITH SAME SESSION ID	
1005	ERROR WHEN CHECKING RPL FROM RCV ANY	
1006	ERROR WHEN BUILDING PWQE	
1008	ERROR WHEN CHECKING RPL FOR NEG RESPONSE	
1009	ERROR WHEN CHECKING RPL FROM SENDING NEG RESP	
1010	RU RECEIVED LARGER THAN MAX BUFFER SIZE	
1011	BUFFER PUT ERROR - PROBABLE BUCB OVERLAY	
1001	ERROR WHEN GETTING DATA FROM BUFFER	PS0RECV Receive Req
1002	ERROR WHEN SETTING UP RPL	PS0SEND
1002	ERROR WHEN ISSUING SEND	Send Request

SNA LU62 for OS/390

The EDBC supports for the IBM SNA LU62 protocol on OS/390. This can be used for OS/390-to-OS/390 access as well as access from other platforms to OS/390.

Requirements

The gateway has been certified for IBM OS/390 SNA LU62 support with the following configuration: ACF/VTAM version 3.2 or higher.

Installation and Configuration

To install SNA LU62 support for the EDBC server, code the following parameters on the stage1 IGWFPSVR statement:

```
TYPE=SNA_LU62
ACB=
NODENAME=
DLOGMODE=
MODETAB=
ENABLE=
```

VTAM ACB and LOGMODE definitions for the SNA_LU62 logical unit are generated and placed in the SAMPLE.CNTL library as members RTIAPPL and RTIMODE. These definitions should be given to personnel responsible for VTAM. The SNA_LU0 interface cannot be started until the VTAMLST and VTAMLIB libraries have been updated with the generated ACB and LOGMODE definitions and the newly defined ACB activated.

Starting and Stopping the SNA_LU62 Interface

The following logical in the ISVREDBC member of EDBC.V2R3.FILES.IIPARM determines whether the SNA_LU62 network interface is to be automatically started during server initialization:

```
II_PROTOCOL_SNA_LU62 = YES;
```

The ENABLE= parameter of the stage1 IGWFPSVR statement determines the value of this logical.

The alternative to starting the interface during server initialization is to activate it dynamically by issuing the following Modify operator command:

```
F EDBC,ACT,PROT=SNA_LU62
```

To stop the interface, issue the following Modify operator command:

```
F EDBC,INACT,PROT=SNA_LU62
```

Note: The above command terminates active SNA_LU62 connections.

Sense Code 08120007 and Possible Loop in VTAM

If you use the SNA LU 6.2 interface with many concurrent users, problems may occur where VTAM rejects incoming BINDS with a sense code of 08120007. This is a VTAM configuration issue. Should this occur, check the NCP gen to ensure that sufficient resources have been allocated by VTAM to support the SNA LU6.2 requirements. You can allocate resources for specific independent logical units by coding the TESSCB parameter on the LU statements defining that particular logical unit. Or a pool of resources can be allocated to be used by any independent logical unit by coding the AUXADDR and ADDSESS parameters on the BUILD statement of the NCP gen. In either case, you should specify values for the parameters that at least equal the desired maximum concurrent users.

KNET TCP/IP for OS/390

The network support for Fibronics KNET OS/390 TCP/IP allows EDBC to communicate with other platforms across a TCP/IP network.

Requirements

The server is certified for KNET OS/390 TCP/IP support with the following configuration: KNET TCP/IP for OS/390 Release 1.3 (or higher).

EDBC uses the KNET API, which implements an SNA interface from the EDBC to the KNET address space. The KNET address space communicates directly with the Fibronics K200 or K2000 controller to provide connectivity to the workstations attached to an Ethernet LAN.

Installation and Configuration

To install TCP_KNET support for the EDBC server, code the following parameters on the stage1 IGWFPSVR statement:

```
TYPE=TCP_KNET  
PLU=  
PORT=  
ENABLE=
```

Starting and Stopping the TCP_KNET Interface

The following logical in the ISVREDBC member of EDBC.V2R3.FILES.IIPARM determines whether the TCP_KNET network interface is to be automatically started during server initialization:

```
II_PROTOCOL_TCP_KNET = YES;
```

The ENABLE= parameter of the stage1 IGWFPSVR statement determines the value of this logical.

The alternative to starting the interface during server initialization is to activate it dynamically by issuing the following Modify operator command:

```
F EDBC,ACT,PROT=TCP_KNET
```

To stop the interface, issue the following Modify operator command:

```
F EDBC,INACT,PROT=TCP_KNET
```

Note: The above command terminates active TCP_KNET connections.

Connecting from a Remote Client

To connect to the EDBC server using the TCP_KNET interface, a vnode entry must be created on the EDBC client. See the “Managing Network Communications” chapter in *EDBC Getting Started* for information on how to create vnode entries for TCP_KNET.

IBM TCP/IP for OS/390

The network protocol support for IBM’s OS/390 TCP/IP product enables EDBC to communicate with other platforms across a TCP/IP network.

Requirements

The gateway is certified for IBM OS/390 TCP/IP support with the following configuration:

- MVS/ESA, OS/390, or z/OS
- IBM TCP/IP for OS/390 V3R2 and above

Installation and Configuration

To install IBM TCP/IP support for the EDBC server, code the following parameters on the stage1 IGWFPSVR statement:

```
TYPE=TCP_IBM  
PORT=  
USERID=  
ENABLE=
```

Starting and Stopping the TCP_IBM Interface

The following logical in the ISVREDBC member of EDBC.V2R3.FILES.IIPARM determines whether the TCP_IBM network interface is to be automatically started during server initialization:

```
IL_PROTOCOL_TCP_IBM = YES;
```

The ENABLE= parameter of the stage1 IGWFPSVR statement determines the value of this logical.

The alternative to starting the interface during server initialization is to activate it dynamically by issuing the following Modify operator command:

```
F EDBC,ACT,PROT=TCP_IBM
```

To stop the interface, issue the following Modify operator command:

```
F EDBC,INACT,PROT=TCP_IBM
```

Note: The above command terminates active TCP_IBM connections.

Connecting from a Remote Client

To connect to the EDBC server using the TCP_IBM interface, a vnode entry must be created on the EDBC client. See the “Managing Network Communications” chapter in *EDBC Getting Started* for information on how to create vnode entries for TCP_IBM.

IBM TCP/IP Problem Diagnosis

The EDBC TCP/IP protocol server issues an error message whenever an IBM TCP/IP error occurs. The format of the error message is as follows:

```
subsysid: TCP-IBM "function" RETCODE = -00000001 ERRNO= error number
```

where:

subsysid: = EDBC subsystem id
function = TCP/IP operation (RECEIVE, SEND, LISTEN, and so on)
error number = TCP/IP error return code

See the *IBM TCP/IP Application Programming Interface Reference* for a complete description of error return codes.

CCI for OS/390

EDBC supports the CCI protocol on OS/390. This is used for OS/390-to-OS/390 access.

Requirements

The gateway has been certified for OS/390 CCI support with the following configuration: CA90s level 9312 or higher.

Installation and Configuration

To install CCI support for the EDBC server, code the following parameters on the stage1 IGWFPSVR statement:

```
TYPE=CCI  
PRODID=  
ENABLE=
```

Starting and Stopping the CCI Interface

The following logical in the ISVREDBC member of EDBC.V2R3.FILES.IIPARM determines if the CCI network interface is to be automatically started during server initialization:

```
IL_PROTOCOL_CCI = YES;
```

The ENABLE= parameter of the stage1 IGWFPSVR statement determines the value of this logical.

The alternative to starting the interface during server initialization is to activate it dynamically by issuing the following Modify operator command:

```
F EDBC,ACT,PROT=CCI
```

To stop the interface, issue the following Modify operator command:

```
F EDBC,INACT,PROT=CCI
```

Note: The above command terminates active CCI connections.

Connecting from a CCI Client

The utility, EDBCNETU, is used to configure a client. The node information for CCI is as follows:

```
Protocol:                CCI
Remote Node Address:     CCI      System ID of other node
Remote Listen Address    CCI      EDBC subsystem ID
```

The Remote Node Address specifies the System ID of the CCI system that you are connecting to. The Remote Listen Address is the Subsystem ID of the EDBC server.

CCI Abend Codes and Messages

Various internal error conditions can cause the CCI interface to abnormally terminate (abend). When an abend occurs, EDBC continues functioning; however, the CCI interface stops functioning and no new connections can be made until the CCI interface is restarted. The abend code and summary information usually appears on the OS/390 console and the gateway JES log. The active load module displayed in the summary information is IICCIPS.

The following table lists the possible abend codes for the CCI protocol server:

Abend	GTF Error Message	Program
1110	RU RECEIVED LARGER THAN MAX BUFFER SIZE	PSCRCVA
1111	BUFFER PUT ERROR - PROBABLE BUCB OVERLAY	Receive any
1202	ERROR WHEN GETTING DATA FROM BUFFER	PSCDFSM receive req

SNS/TCP for OS/390

The network protocol support for SNS/TCP on OS/390 is functionally the same as IBM's OS/390 TCP/IP protocol support. That is, it allows EDBC to communicate with other platforms across a TCP/IP network.

Requirements

The gateway has been certified for SNS/TCP with the following configuration: SNS/TCP Version 2.0 or higher.

Installation and Configuration

To install SNS TCP/IP support for the EDBC server, code the following parameters on the stage1 IGWFPSVR statement:

```
TYPE=TCP_SNS  
APPLID=  
PORT=  
SYSID=  
ENABLE=
```

Starting and Stopping the TCP_SNS Interface

The following logical in the ISVREDBC member of EDBC.V2R3.FILES.IIPARM determines whether the TCP_SNS network interface is to be automatically started during server initialization:

```
II_PROTOCOL_TCP_SNS = YES;
```

The ENABLE= parameter of the stage1 IGWFPSVR statement determines the value of this logical.

The alternative to starting the interface during server initialization is to activate it dynamically by issuing the following Modify operator command:

```
F EDBC,ACT,PROT=TCP_SNS
```

To stop the interface, issue the following Modify operator command:

```
F EDBC,INACT,PROT=TCP_SNS
```

Note: The above command terminates active TCP_SNS connections.

Connecting from a Remote Client

To connect to the EDBC server using the TCP_SNS interface, a vnode entry must be created on the EDBC client. See the “Managing Network Communications” chapter in *EDBC Getting Started* for information on how to create vnode entries for TCP_SNS.

Enable and Test Security Interfaces

EDBC interfaces to IBM’s Resource Access Control Facility (RACF) and the Computer Associates’ Access Control Facility (CA-ACF2) and Top Secret Security (CA-TSS) security products. The security interface is activated by the following logical in the ISVREDBC member of EDBC.V2R3.FILES.IIPARM:

II_SECURITY =

The SECURITY = parameter of the stage1 IGWFINET statement determines the value of this logical. The options are RACF, ACF2, TSS or NONE.

The interface to the security products occurs in an exit routine that is invoked when a user attempts to connect to EDBC. The exit routine in turn calls the security product to authenticate the user. There is an exit routine for each of the supported security products.

The sources for these routines are distributed in the FILES.ASM library as members IIRACF, IIACF2, and IITSS.

IBM Resource Access Control Facility (RACF)

EDBC supports IBM’s Resource Access Control Facility (RACF). This facility performs authentication checking and resource control for an OS/390 system. It requires that a remote client be authenticated by RACF before a connection is established.

When RACF security is requested, the EDBC server uses RACF to validate the user ID and password. This validation occurs during connection processing.

If the validation fails, then the connection request is rejected and RACF issues message ICH408I. If the validation of the user ID/password is successful, then RACF returns a pointer to a security control block.

The interface to RACF uses the IBM RACINIT interface to issue the logon validation requests from EDBC clients to RACF.

Installing and Customizing the RACF Interface

This section describes how to customize EDBC to enable RACF security. Complete the following steps:

Install the EDBC server.

Create a RACF profile for the EDBC server.

1. Create a RACF profile for each EDBC client.
2. Define the user created in the RACF profile to EDBC.
3. Create a vnode definition on the EDBC client.
4. Set the II_SECURITY logical symbol.
5. Test the RACF interface.

Creating a RACF Profile for the Server

Create a RACF identifier for the EDBC task.

Creating a RACF Profile for Each Gateway Client

Create a RACF profile for each user who will access the server. This profile should have the same characteristics as that of a TSO user.

Defining the User to EDBC

The user created in the RACF profile definition must be defined to the server. See the “Maintaining the Gateway” chapter for instructions on adding new users.

Creating a Vnode on the Client

On the client use the EDBC Network utility to create a vnode entry.

See *EDBC Getting Started* for instructions on creating vnodes.

Setting the II_SECURITY Logical Symbol for the EDBC Server

To enable the RACF security interface, add the following logical symbol to the ISVREDBC member in EDBC.V2R3.FILES.IIPARM.

```
II_SECURITY = RACF;
```

Testing the RACF Interface

Use the following procedure to test the RACF interface:

1. Start the EDBC server.
2. On the EDBC client, invoke the EDBC Network utility.
3. Right-click the newly created vnode and select SQL Test.
4. If the SQL test panel appears, the RACF interface has been successfully installed.

Computer Associates Access Control Facility 2

The server supports the Computer Associates Access Control Facility 2 (CA-ACF2). This facility performs authentication checking and resource control for the OS/390 system. It ensures that a remote client is authenticated before a connection is established. If the user ID/password fails, then the remote connection request is rejected. If the user ID/password is valid, the connection processing continues.

The interface to CA-ACF2 uses the IBM Security Access Facility (SAF) to issue the logon validation requests from the EDBC client to CA-ACF2.

Installing and Customizing the CA-ACF2 Interface

To enable the CA-ACF2 security interface, complete the following steps:

1. Create a CA-ACF2 Logon Identifier (LID) for the server.
2. Create a CA-ACF2 protection record.
3. Create a CA-ACF2 LID for each user.
4. Define the user to the server.
5. Create a vnode for each EDBC client.
6. Set the IL_SECURITY logical symbol for the server.
7. Test the CA-ACF2 interface.

Creating a Server
CA-ACF2 LID

Create a CA-ACF2 identifier for the EDBC task. It should have the same characteristics as CICS. Verify that the following fields are set:

- MUSASS
- STC

Creating a CA-ACF2
LID for Each User

Create a CA-ACF2 LID for each user who will access the server. This LID should have the same characteristics as that of a TSO user.

Defining the User to
EDBC

The user created in the RACF profile definition must be defined to the server. See the "Maintaining the Gateway" chapter for instructions on adding new users.

Creating a Vnode on
the Client

On the client use the EDBC Network utility to create a vnode entry.

See *EDBC Getting Started* for instructions on creating vnodes.

Setting the
II_SECURITY Logical
Symbol for the EDBC
Server

To enable the CA-ACF2 security interface, add the following logical symbol to the ISVREDBC member in the EDBC.V2R3.FILES.IIPARM.

```
II_SECURITY = ACF2;
```

Testing the CA-ACF2
Interface

Use the following procedure to test the CA-ACF2 interface:

1. Start the EDBC server.
2. On the EDBC client, invoke the EDBC Network utility.
3. Right-click the newly created vnode and select SQL Test.
4. If the SQL test panel appears, the RACF interface has been successfully installed.

Computer Associates Top Secret Security Facility

The server supports the Computer Associates Top Secret Security (CA-TSS). This facility performs authentication checking and resource control for an OS/390 system. It ensures that a remote client is authenticated before a connection is established. If the user ID/password fails, the remote connection request is rejected. If the user ID/password is valid, then the connection processing continues.

The interface to CA-TSS uses the IBM RACINIT to issue the logon validation requests from the server to CA-TSS.

Installing and Customizing the CA-TSS Interface

To enable CA-TSS security, complete the following steps:

1. Create a CA-TSS Facility for the server.
2. Create a CA-TSS Access Control Identifier (ACID) for the server.
3. Create a CA-TSS ACID for each user.
4. Define the user to the server.
5. Create a vnode on the EDBC client.
6. Set the II_SECURITY logical symbol for the EDBC server.
7. Test the CA-TSS interface.

Creating a Server
CA-TSS Facility

A CA-TSS facility must be created to authorize the server to issue RACINT requests.

The following example shows the suggested control options for the server:

```
GATEWAY
INITPGM=IIG ID=D TYPE=31
ATTRIBUTES=ACTIVE, SHRPRF, NOASUBM, NOTENV, NOABEND,
ATTRIBUTES=MULTIUSER, NOXDEF, LUMSG, STMSG, SIGN(M),
ATTRIBUTES=NOPSEUDO, INSTDATA, NORNDPSW, AUTHINIT,
ATTRIBUTES=NOPROMPT, NOMENU, NOAUDIT, NORES, NOMRO,
ATTRIBUTES=WARNPW, NOTSOC, NOTRACE, NOLAB, NOEXTEND,
ATTRIBUTES=NODORMPW, NONPWR, NODATACOMXTND
MODE=FAIL
LOGGING=ACCESS, INIT, SMF, MSG
UIDACID=8 LOCKTIME=000 DEFACID=*NONE* KEY=8
```

Creating a Server
CA-TSS ACID

Create a CA-TSS identifier for the EDBC task. This Access Control Identifier (ACID) should be associated with the OS/390 started task. In this example, the ACID name is EDBC1. The ACID must be able to access the server CA-TSS facility and the Vantage CA-TSS facility.

Creating a CA-TSS
ACID for Each User

Create a CA-TSS ACID for each user who will access the server. This ACID should have the same characteristics as that of a TSO user.

The client must use TSO to set the correct password for this authorization ID. The gateway does not support the setting of the CA-TSS password from a gateway client.

Defining the User to
EDBC

The user created in the CA-TSS profile definition must be defined to the server. See the "Maintaining the Gateway" chapter for instructions on adding new users.

Creating a Vnode on
the Client

On the client use the EDBC Network utility to create a vnode entry.

See *EDBC Getting Started* for instructions on creating vnodes.

Setting the
II_SECURITY Logical
Symbol for the EDBC
Server

To enable the CA-TSS security interface, add the following logical symbol to the ISVREDBC member in EDBC.V2R3.FILES.IIPARM.

```
II_SECURITY = TSS;
```

Testing the CA-TSS
Interface

Before you test the CA-TSS interface, you should have successfully completed installing the gateway.

Use the following procedure to test the CA-TSS interface:

1. Start the EDBC server.
2. On the EDBC client, invoke the EDBC Network utility.
3. Right-click the newly created vnode and select SQL Test.
4. If the SQL test panel appears, the RACF interface has been successfully installed.

Force Inactivate Timeout

The force inactivate timeout facility cleans up threads that are hung as a result of circumstances outside the control of the server. The facility frees up user threads after a pre-specified period of time, following the failure of a disconnect or inactivate. It cannot be invoked directly by the user, but must first be preceded by the failure of a user disconnect, an operator inactivate, or an inactivate user timeout.

To activate the force inactivate timeout facility, set the following symbol in EDBC.V2R3.FILES.IIPARM(ISVREDBC).

```
II_FORCE_TMOUTINT = nn;
```

where *nn* is the number of minutes that elapses before a force inactivate is executed against the thread that did not successfully inactivate or disconnect. The default is five minutes.

If the parameter is not specified, or if it is set to 0, the inactivate user timeout facility is not activated and therefore, no force inactivates occur. Error messages normally accompany a force inactivate timeout.

Note: Do not confuse the force inactivate timeout logical with the SRV_TMOUTINT or SRV_TMOUTINT or II_INACTV_TMOUTINT logicals. SRV_TMOUTINT controls how often all of the timers (including the force inactivate timer) are checked. II_INACTV_TMOUTINT controls the length of time a user session can have no activity before it is automatically inactivated. For more information on these logicals, see the “Server Logical Symbols and the IIPARM Clist” appendix.

You may also set the II_FORCE_TMOUTINT symbol during stage2 of the IIVP.

Local Time Zone

The EDBC server logical II_TIMEZONE is used to specify the difference in hours between Greenwich mean time (GMT) and the local time zone where the EDBC server is running. Specify this logical in EDBC.V2R3.FILES.IIPARM(ISVREDBC) as follows:

```
II_TIMEZONE = 'n';
```

where *n* specifies the local time zone displacement. If this parameter is not specified, it defaults to 0.

When the EDBC server is initialized, the value of the `II_TIMEZONE` logical is compared to the system GMT time zone displacement. If you have defined the logical to match the value of your local time zone, you will see the following message at startup time:

```
EDBC: II_TIMEZONE set to n
```

If the value you set for `II_TIMEZONE` does not match the value of your local time zone as set in `SYS1.PARMLIB(CLOCKxx)`, you will see the following message at startup time:

```
EDBC: II_TIMEZONE set to n
EDBC: II_TIMEZONE (n) does not match local timezone(n)
EDBC: II_TIMEZONE DATE conversion may be incorrect
```

If `II_TIMEZONE` is defined incorrectly on either the server or client installation, dates will appear incorrect.

The gateway does not support the `II_DATE_FORMAT` logical; therefore, all date input strings must be in US format. See the *EDBC OpenSQL Reference Guide* for a list of valid US date formats.

Year 2000 Support

The installation accommodates the year 2000 by allowing you to set the `II_DATE_CENTURY_BOUNDARY` logical symbol. See the “Server Logical Symbols and the IIPARM Clist” appendix for more information about this logical symbol.

Alternate Translation Tables

The server can use alternate translation tables that allow the default table (Country Extended Code Page 037 for the USA) to be overridden.

Note: This capability only applies to single byte character sets.

To override the default table, use the `IIPARM` logical, `II_GCC_TRANSLATE_TABLE`, which specifies the name of the alternate translation table to be used. For example:

```
II_GCC_TRANSLATE_TABLE = 'CECP277';
```

The above example would bring in the Denmark/Norway translate table. Some, but not all, alternate translation tables are supplied with the server. Members `CECPnnn` (where `nnn` is Extended Code Page number) are supplied in the `FILES.ASM` library in source form and the compiled forms are in the `BACK.LOAD` library.

A job (member CECPNNN in the SAMPLE.CNTL library) can be run, with modifications, to compile an alternate translation table.

The translate table causes the server to change incoming (GCC Network Standard single byte characters [ASCII with extensions]) characters into local EBCDIC characters. (A reverse table is built for outgoing characters.)

To build a new table, start with the USA table CECP037 and make changes where appropriate. The notes in the CECP037 table will assist you with the needed changes. In summary, there are four types of changes:

- Code Page Name – CECPnnn where nnn is the CECP identifier.
- Tilde substitute value – the hex value of whatever server value x'7E' is translated to.
- Dollar Sign substitute value – the hex value of whatever server value x'24' is translated to.
- Server substitute values (multiple) – in the CECP037 source, each server value that translates into a non-zero value will indicate, in comments, what the server value represents. All of these should be reviewed and changes made as required.

When an OS/390 client connects to an EDBC server, the server must force the client to translate its outgoing messages. This is accomplished by including member FORCEHET in the IIPARM concatenation in the server JCL.

Setting Up Vantage Gateway Users

The gateway DBA must create an environment for each user who accesses the gateway before the user can connect to the gateway. This chapter describes how to do this and the concepts behind these procedures.

Gateway User Setup Summary

As the gateway DBA, you must create an environment for each user who accesses the gateway. To do so, complete the following tasks for each user:

1. Define an OS/390 user ID with authorized access to TSO.
2. Define the user to the gateway AREA that he or she will be accessing. If the new user will be accessing an existing AREA, perform the following:

Using the Terminal Monitor (local or remote), connect to the IIDBDB partition in the gateway AREA to which the user is to be defined and issue the following SQL statement:

```
INSERT INTO IIUSER VALUES ('newuser',0,0,32789);
```

where 'newuser' is the TSO user ID of the new user.

If the new user will be accessing a new gateway AREA, perform the following:

Using the TSO editor, add the user(s) to the EDBC.VANT.FILES.USERS.DATA data set.

Use the IGWFDBA ISPF utility to create and format the new gateway AREA. The formatting of the AREA adds the IIDBDB partition to the new gateway area and populates the IIUSER table with all of the users defined in the EDBC.VANT.FILES.USERS.DATA data set.

3. Add ING_AREA_newuser and ING_SORT_newuser logicals to the EDBC server IIPARM and recycle the server.
4. Check access to each partition from the Terminal Monitor, running locally under TSO.
5. Check access to each partition from a remote client.

After you create the OS/390 user IDs, you perform the remaining tasks using the IGWFDBA utilities that were introduced in the “Using the IGWFDBA Utilities” chapter. This chapter provides step-by-step instructions for each of these tasks.

Expanding the Vantage User Base

The Vantage gateway expands the community of Vantage users to include EDBC users working on Microsoft Windows, IBM OS/2, or UNIX platforms.

In many ways, supporting gateway users is similar to supporting other Vantage users. However, supporting gateway users also involves tasks that are specific to the gateway. These tasks are described in the following sections:

- Define the needs of the gateway user
- Registering Vantage data

Defining Needs of Gateway Users

The gateway DBA must identify the data the gateway user wants to see, and determine whether it is appropriate to authorize access to this information. Note that the Vantage gateway provides *read* and *write* access to Vantage data, however the initial release of the Vantage interface will only support *read* access requests.

In addition, you need to understand the types of queries the user will be running. A user who will run simple select statements against a given group of files has standard gateway storage needs. A user who will run complex queries such as three- or four-way join definitions against Vantage data may require a gateway area that is larger than the default size.

Registering Vantage Data

Vantage data must be registered in order for a gateway user to access it through the gateway. See the “Registering Vantage Data” chapter for full details about the registering process.

Concepts: Area, Sort Data Set, Partition, and Driver

This section presents certain concepts that are basic to the gateway architecture.

Area and Sort Work Data Sets

Each gateway user must be assigned to a gateway area to support access to Vantage. A user can be assigned to a dedicated gateway area or share an area with other users. The default size for an area is 2700 4 KB records. This makes approximately 22 MB available for the storage of gateway objects. The maximum size for an area is 256 MB. A maximum of 70 partitions per area is supported.

Each user also requires the pre-allocation of a sort work data set. Like the gateway area, the sort work data set can be dedicated to a user or shared. This enables the gateway DBA to control the flow of DASD allocation for each user ID. If the user ID requires standard sorting, use the default 4 KB records. A secondary allocation equal to the primary space is the default.

Vantage allows up to 119 extents. The sort data set will be dynamically extended by the gateway if a secondary space allocation exists.

Each user requires a set of logical symbols that connect the user ID to the appropriate area and sort data set. The location of these logical symbols varies, depending on whether the user is accessing the gateway area locally or remotely. For remote access, the user's logical symbols are stored in the member with the default name DBNMEDBC, in the library with the default name EDBC.V2R3.FILES.IIPARM. For local access to the gateway (from TSO, without going through the name server) the user's logical symbols are stored in the SABExxxx member or the Uxxxxxxx member of the EDBC.V2R3.FILES.IIPARM library.

Partition

Each gateway area contains one or more partitions. Partitions are suballocated from within an area. Each partition represents a database to the Vantage gateway user.

Each partition stores a complete set of gateway catalogs, which are required for relational processing.

The partition stores table templates for all the files that have been registered in that partition. It also stores any application objects, such as forms or reports that have been created for use with the data in this partition.

Note: The partition does *not* store the actual data for the registered files. The data is accessed from Vantage each time it is queried. The table templates and application objects reside in the partition; the data they reference does not.

Using the IGWFDBA Utilities

Use the IGWFDBA utilities to set up a Vantage gateway user. These utilities provide skeleton JCL you can customize and store for use later.

Adapting the Default Profile for a New User

The profile panel enables you to predefine a series of variables that the IGWFDBA utilities use to build the JCL for the user. This procedure allows you to customize the profile for the user in question. If you do not adapt the profile as described here, you have to edit the JCL manually to incorporate the necessary values, such as the proper user ID and the name of the gateway area that is being created for that user.

To predefine the variables:

1. Invoke the IGWFDBA utilities. See The IGWFDBA Main Menu section in the “Using the IGWFDBA Utilities” chapter for more information.
2. Enter 0 in the Selection field and press Enter to display the EDBC/Gateway Facility Profile Panel.

This panel shows default values that were customized by the IIVP Stage 2-installation process.

An example of this screen is shown in the following panel:

```

----- EDBC/Gateway Facility Profile Panel -----
IIPARM  => ISUR(EDBC) PREFIX(EDBC.USAMGW) SABE(DEMO)
INGAREA => EDBC.USAMGW.INGAREA.CAIDEM1
INGJCL1 => //IGWFJOBS JOB (CAIDEM1),NOTIFY=CAIDEM1.CLASS=A
INGJCL2 => //MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=CAIDEM1,REGION=6M
INGJCL3 => //***
INGJCL4 => //***
INGPREFIX => EDBC.USAMGW
IIPREFIX =>
INGMODGN =>

INGISISUR => ISUREDBC   INGSSABE => SABEDEMO
INGDRIUR => LOCAL      EDCUSER  => CAIDEM1   INGAUTH => N
INGULS   => MUSUOL     INGSYSDA => UIO       INGUNT  => 3380

IIIMS    =>           IIMSID   =>
IIUPDATE => -U        IIMAXC  => 999
INGARPRI => 2700     INGARSEC => 540
INGSRPRI => 150      INGSRSEC => 150
INGDB2IS =>         INGDB2AD =>

F1=HELP   F2=SPLIT   F3=END     F4=RETURN  F5=RFIND   F6=RCHANGE
F7=UP     F8=DOWN    F9=SWAP    F10=LEFT   F11=RIGHT  F12=RETRIEVE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
3/18 | CAPS | 14:12:50 | IBM-3278-2

```

Customize the parameters as required. See the Parameters on this Panel section in the “Using the IGWFDDBA Utilities” chapter for a detailed description of each parameter.

Defining the Logical Symbols for the Gateway User

This procedure creates the logical symbols that allow a gateway user to access a specified area and sort work data set. Using the ISPF EDIT option, you create or update the appropriate member of the EDBC.V2R3.FILES.IIPARM logical symbols library.

To create the logical symbols for the user:

1. For remote users, or for local users who will access the Vantage gateway via the EDBC server (server mode):
 - a. Edit the DBNM member of EDBC.V2R3.FILES.IIPARM. With the default prefix, the file name has the form *EDBC.V2R3.FILES.IIPARM(DBNMxxxx)*, where xxxx is the EDBC server ID.
 - b. Add entries to this file to specify the user’s ING_AREA and ING_AREA and ING_SORT logicals.

A sample member is shown below, with the default identifier EDBC:

```

/*          DBNMEDBC                      */
/*  SETUP THE DATABASE LOCATIONS FOR NON-RELATIONAL
    GATEWAY USER */
/*          */
ING_AREA_INGDBA=EDBC.V2R3.INGAREA.DEMO ;
ING_SORT_INGDBA=EDBC.V2R3.INGSORT.DEMO ;
ING_AREA_EDCUSR1=EDBC.V2R3.INGAREA.DEMO ;
ING_SORT_EDCUSR1=EDBC.V2R3.INGSORT.DEMO ;
ING_AREA_EDCUSR2=EDBC.V2R3.INGAREA.DEMO ;
ING_SORT_EDCUSR2=EDBC.V2R3.INGSORT.DEMO ;

```

- c. Save the updated member.
2. For all users who will access the gateway outside of the EDBC server (standalone mode):

Note: The Format Area JCL generated by the IGWFDBA utility will perform this step automatically.

- a. Edit a new member of EDBC.V2R3.FILES.IIPARM.

With the default prefix, the file name has the form *EDBC.V2R3.FILES.IIPARM(Uxxxxxxxx)*, where xxxxxxxx is the user ID.

- b. Add entries to this file to specify the EDBC server, and the user's ING_AREA and ING_SORT logicals. Specify the ING_MODE parameter as NOSERVER. A sample member is shown below, with the default identifier, UCAIDEM1:

```

/*          */
/*  RUN INGRES IN STANDALONE MODE - UCAIDEM1 */
/*          */
II_DBMS_SERVER = EDBC ;
II_GCN_ID=EDBC;
ING_AREA=EDBC.V2R3.INGAREA.DEMO
ING_SORT=EDBC.V2R3.INGSORT.DEMO
ING_MODE =      NOSERVER ;

```

- c. Save the new member.

Formatting a Gateway User's Area

Use the following procedure to format an area for the Vantage gateway user and build the required sort work data set:

1. Invoke the IGWFDBA utilities. See the IGWFDBA Main Menu section in the "Using the IGWFDBA Utilities" chapter for more information.
2. From the EDBC Gateway Facility Database Administrator Main Menu (IGWFDBA) panel, type 2.7 at the Selection line and press Enter.

An example of this screen is shown in the following panel:

```

tpx.cai.com
----- EDBC/Gateway Create JCL to FORMAT an EDBC Area -----
                                USERID - MURTO01
                                TIME   - 14:14

ENTER THE REQUIRED PARAMETERS:

Server => ISUREDBC      /*Logical symbol of EDBC Server */
Ingarea => SABEDEM0     /*Logical symbol of User Area  */
Userid  => CAIDEM1      /*EDBC logical userid         */
Ingarea DSName => EDBC.USAMGW.INGAREA.CAIDEM1 /*EDBC Area dataset name     */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE
F7=UP        F8=DOWN       F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEU

ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
10/19 | CAPS | 11:18:29 | IBM-3278-2
    
```

3. Check the screen to see that the values for the following parameters are correct:

- | | | |
|-------------------|----|--|
| SERVER | => | The name of the server logical symbols member in <i>prefix.FILES.IIPARM</i> .
For example: ISVREDBC |
| INGAREA | => | The logical symbol member that associates the given user ID with a given area and sort work data set.

This logical symbols member is stored in <i>prefix.FILES.IIPARM</i> . |
| USERID | => | The user ID to whom the area is assigned. |
| INGAREA
DSNAME | => | The name of the area data set you are formatting.
Enter the fully qualified name of the data set. |

4. To advance to the skeleton JCL, press Enter.

The temporary JCL appears.

5. Review and customize the JCL as required.

See the Gateway Partitions section for a discussion of how you may need to customize the JCL.

6. Submit the JCL to allocate and format the gateway area.
All return codes should be 0 or 4. The last step should be 0.
7. When the area is formatted, press PF3 to return to the EDBC/Vantage Gateway Facility Main Menu.

Gateway Partitions

The gateway data manager requires that a Vantage gateway user connect to the database that stores the data being accessed. For purposes of the Vantage gateway, the gateway database is equivalent to the gateway partition. Where a user would normally connect to a specific database, the gateway user instead connects to the specified gateway partition. Accordingly, you must inform the user of the name of any gateway partition that is made available for his/her use. The user does not need to know the name of the user area or the data set prefix, but the partition name represents crucial routing information.

Gateway Catalogs and the Gateway Partition

A partition is a working area for a gateway user. A partition is created within a gateway area. The utility that creates the partition also updates the gateway catalogs that support relational queries for the partition.

Creating a Partition

To create a partition for a Vantage gateway user:

1. Invoke the IGWFDDBA utilities. See The IGWFDDBA Main Menu section in the “Using the IGWFDDBA Utilities” chapter for more information.
2. From the EDBC Gateway Facility Database Administrator Main Menu (IGWFDDBA) panel, type **2.8** at the Selection line and press Enter.

The terminal displays a screen like the one shown in the following panel:

```

----- EDBC/USAM Create JCL to CREATE a User Partition -----
                                USERID - MURTO01
                                TIME   - 14:15

ENTER THE REQUIRED PARAMETERS:

Partition => IIUPUSM      /*Partition for USAM Database */
Server    => ISUREDBC    /*Logical symbol of EDBC Server */
Ingarea   => SABEDEM0    /*Logical symbol of User area */
Usam Driver => LOCAL     /*Specify USAM driver id */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=RFIND   F6=RCHANGE
F7=UP        F8=DOWN     F9=SWAP    F10=LEFT   F11=RIGHT  F12=RETRIEU
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----
                                10/22 | CAPS | 11:19:21 | IBM-3278-2

```

3. If necessary, type in the appropriate values for the parameters:

PARTITION => User partition name for the Vantage Database.

SERVER => Name of server logical symbols member in the *prefix.FILES.IIPARM*.

For example: ISVREDBC

INGAREA => The logical symbol member for the user in question. This logical symbols member is stored in *prefix.FILES.IIPARM*.

4. To display the JCL, press Enter.
5. Review and customize the JCL as required.
See the Customizing the Skeleton JCL to Create a Partition section in the "Setting Up Vantage Gateway Users" chapter for a discussion of how you may need to customize the JCL.
6. Check for a return code 0 when the job is completed.
7. When the partition is built, return to the EDBC/Vantage Gateway Facility Main Menu by pressing PF3.

Checking Access to the Partition

After you have created the partition and built and activated the driver, verify that you can use the driver to access the partition from TSO. To access a gateway partition from TSO:

1. Invoke the IGWFDBA utilities. See the IGWFDBA Main Menu section in “The IGWFDBA Main Menu” chapter for more information.
2. From the EDBC Gateway Facility Database Administrator Main Menu, type **2.5** at the Selection line and press Enter to invoke the Terminal Monitor facility.

The terminal displays a screen similar to the one shown below:

```

tpx.cai.com
----- EDBC/GUTM Gateway Using Terminal Monitor -----
COMMAND ==>                                USERID - MURTO01
                                           TIME   - 14:06
                                           SYSTEM - XAE1

==> SQL iipusm/usam

----- IIPARM profile parameters -----
| INGPREFIX ==> EDBC.USAMGW
| INGSISUR  ==> ISUREDBC
| INGSSABE  ==> SABEDEM0
-----

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to process )

F1=HELP      F2=SPLIT    F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
F7=UP        F8=DOWN      F9=SWAP     F10=LEFT    F11=RIGHT   F12=RETRIEU
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
5/11 | CAPS | 11.10.09 | IBM-3278-2
    
```

3. Check the name of the partition and the server class.

The server class should always be Vantage. The IIPARM profile parameters should reflect the parms you allocated above. If you wish to change any of these values, type in the appropriate entries, which are then automatically allocated.

4. Press Enter to accept the contents of the screen.

If you allocated the parms correctly, the screen displays the Terminal Monitor banner, then the * prompt, which shows that you are in the Terminal Monitor.

5. To display a small table for this partition, type:

```
select * from iidbconstants\g
```

then press Enter.

6. To exit from the Terminal Monitor, type:
 \q
 then press Enter.
7. To exit from IGWFDBA, press PF3 until you return to ISPF.

Registering Vantage Data

Vantage data is stored in special relational tables (called *objects*) maintained by active Vantage subsystems. Registering a Vantage object enables a properly authorized EDBC client to access the data in this object and work with it in the form of a relational table.

Registration information is stored in standard catalogs in the gateway user's area. These catalog entries identify the Vantage subsystems and how their objects should be presented in relational form. The catalogs store the templates for gateway tables. Each time an EDBC client user accesses a table through the gateway, the data is accessed from Vantage. This means that the EDBC client always sees the current version of the data.

This chapter describes the procedure by which the gateway DBA registers data using the VANTEDBC utility to issue the necessary SQL register table commands. In addition, the chapter provides detailed information on the parameters of the register table command.

Vantage Objects and Relational Tables

The Vantage gateway emulates a VSAM KSDS access method to access Vantage objects. This means that Vantage objects may be accessed using a keyed query. Standard VSAM KSDS access assumes that the key is made up of one or more contiguous fields within the KSDS records, whereas Vantage allows for keyed access based upon every field in its objects.

To properly map EDBC tables to Vantage objects as it relates to keyed access, only one Vantage object field may be defined to EDBC as a key field.

Registration

Registration is a mapping process that specifies:

- The association of specific Vantage entities with relational tables
- The layout of the fields within the Vantage record

These fields have characteristics such as length and type of data. In the corresponding relational table, the field characteristics are defined in terms of column names and EDBC data types. It is not necessary to define all of the fields in a Vantage record. Fields may be skipped by the use of an OFFSET clause

- The association of Vantage keys with their EDBC equivalents

Registering Vantage Objects

Registering
Predefined Vantage
Objects

The gateway can be used only to register a predefined Vantage object. In order to use the register table statement, the Vantage object must first be created and available to Vantage. If a Vantage object does not exist within the Vantage subsystem at the time you run the registration statement, the registration will fail.

SQL Register Table Statement

When you register a Vantage object, you define it as a relational table by providing the following information:

- A name for the table
- A name for each column in the table
- The EDBC data type for each column in the table
- The name of the Vantage object that is the source for the table
- The type of DBMS in which the data is stored (must be VSAM for Vantage objects)
- The storage structure for the relational table
- The key for the Vantage object
- The approximate number of rows in the table
- Whether the table can accept duplicate rows
- Whether updates are logged to a journal file

- Whether the gateway should attempt recovery from a system crash
- Whether users can update rows in the table through the gateway (updates are not supported in the initial release)

The full syntax of the register table statement is:

```
register table tablename
  (col_name format [is 'offset()'],
  (col_name format [is 'ext_format']
  {col_name format [is 'ext_format']})
as import
  from 'source'
with
  dbms = {vsam}
  [structure = sortkeyed]
  [key = (col_name [asc | desc]

  [rows = nnnnnnnn]
  [[no]duplicates]
  [[no]journaling]
  [[no]recovery]
  [[no]update]
```

See the "Introduction" chapter of this guide for a description of the conventions used to specify this syntax. The following table outlines the parameters for the register table command. The Descriptions of Register Table Parameters section describes the parameters in more detail.

Parameter	Description	Status
<i>tablename</i>	The relational table name for the Vantage object being registered. This name must follow the gateway naming conventions, as described in this chapter.	Required
<i>col_name</i>	The list of fields in the table.	Required
<i>format</i>	A valid EDBC data type.	Required
is ' <i>offset()</i> '	Used to skip fields in the record. The value specified is the number of bytes relative to 0 from the start of the record. This clause must be specified for the first field after a skipped field. Fields are assumed to be adjacent if this clause is not specified.	Optional

Parameter	Description	Status
is <i>'ext_format'</i>	If the data in the gateway object differs from the EDBC data type, <i>'ext_format'</i> must be specified. Each gateway data object has specific valid external formats.	Often required
as import	Indicates that you are registering a gateway data object.	Required
from <i>'source'</i>	Vantage – name of Vantage object. The Vantage object name must be 9-characters, padded to the right with blanks, followed by the string, PARM=VIO0. This indicates that the EDBC exit 0 should be invoked to handle requests against this object. Exit 0 should be set to the VANTIO load module, which implements the EWS interface into Vantage.	Required
with	Provides additional information about the object being imported.	Required
dbms	Specifies the gateway with which this table is being registered. The only valid value for Vantage objects is VSAM.	Required
structure	Defines the type of keyed searches, if any, that are allowed on the gateway object. For Vantage objects, specify sortkeyed.	Optional
key	Specifies the column that makes up the primary key in a keyed search. Columns must be specified in either ascending or descending order. The default is ascending.	Optional
rows = <i>nnnnnnnnn</i>	Specifies the approximate number of rows in the table. Since the optimizer uses this value when building query plans, it is recommended that the user provide an initial count of rows. The default is 1000.	Optional

Parameter	Description	Status
[no]duplicates	Indicates whether the table can contain duplicate rows. Due to the nature of For Vantage, it is recommended that you allow for duplicate rows.	Optional
[no]journaling	Specifies whether updates are logged to the journal files. For the Vantage gateway, you must specify nojournaling, which is the default.	Optional
[no]recovery	Specifies whether the gateway should have exclusive control over the gateway object and attempt to recover from a system crash. For the Vantage gateway, you must specify norecovery, which is the default.	Optional
[no]update	Specifies whether the table may be updated. The default is the update intent of the partition. However, updates are not supported in the initial release of Vantage tables, and therefore noupdate should be specified.	Optional

Descriptions of Register Table Parameters

This section gives detailed descriptions for those register table parameters that require further explanation.

Relational Table Names

Each relational table must have a name that is unique within the gateway partition.

The table name will appear in the standard system catalogs as a table. The table name does not imply what type of object—EDBC, Vantage—the table is.

Observe the following rules and conventions:

- The table name can be up to 24 characters in length.
To ensure application portability, however, limit names to 18 characters.
- The first character of the name must be alphabetic (a-z) or

an underscore (_).

You can use the characters 0–9, @, or \$ in the name after the first character.

- Names are case insensitive

Any uppercase alphabetic character is converted to lowercase.

See the *EDBC OpenSQL Reference Guide* for naming rules and for a list of the reserved words that cannot be used as names for tables or indexes.

Column Descriptions

Each field to be referenced must be assigned a column name and an EDBC data type. If the data type does not match the external (file) format, then an “is ext_format” clause must be specified. The “is offset ()” is used to skip fields which will not be referenced.

The rules for naming columns are the same as those for naming tables.

The gateway performs the necessary translations as data passes through it, and rejects any conversions it cannot perform. This is generally detected during parsing.

Data Type Mapping

Precise mapping between the data types for Vantage objects and EDBC data types is crucial to the operation of the gateway. The gateway has no method of verifying the data types you assign. If you map a COBOL data type to an EDBC data type that does not correspond, you may or may not receive an error message, but the results of the query are necessarily erroneous.

The tables in the following sections show the EDBC data types that correspond to data types in COBOL, PL/I, and 370 Assembler. The characteristics of the EDBC data types are described more fully in the sections that follow the tables.

Be aware that, in converting to an EDBC data type, the gateway may truncate data if the original data field in Vantage is longer than the EDBC data type to which it is converted. Data conversion may also result in a loss of precision.

The following table shows the COBOL and EDBC data types:

COBOL Data Type	EDBC Data Type	EDBC Range
display	char	1 to 2000 char
numeric edit	char	1 to 2000 char

COBOL Data Type	EDBC Data Type	EDBC Range
zoned numerics	char (for display only, not for computation) float4 (4-byte) (maximum of seven digits) float (8-byte) (maximum of 15 digits) integer (4-byte) (maximum of 9 digits)	1 to 254
binary halfword COMP or COMP-4 (1 to 4 digits)	smallint (2-byte)	-32,768 to +32,767
binary fullword COMP or COMP-4 (5 to 9 digits)	integer(4) (4-byte)	-2,147,483,648 to +2,147,483,647
binary doubleword COMP or COMP-4 (10 to 18 digits)	Not supported	
display float (external float)	Not supported	
internal float COMP-1 4 bytes	float4 (4-byte)	-1.0e+38 to +1.0e+38 (7-digit precision)
internal float COMP-2 8 bytes	float (8-byte)	-1.0e+38 to +1.0e+38 (15-digit precision)
packed decimal COMP-3 (external format required – decimal or packed decimal)	float4 (4-byte) (maximum of 7 digits) or	-1.0e+38 to +1.0e+38 (7-digit precision)
	float (8-byte) (maximum of 15 digits)	-1.0e+38 to +1.0e+38 (15-digit precision)
	integer (4-byte) (maximum of 9 digits)	-2,147,483,648 to +2,147,483,647

The following table shows the PL/I and EDBC data types:

PL/I Data Type	PL/I Range	EDBC Data Type	EDBC Range
character (n)	$1 \leq n \leq 254$	char	1 to 254 char
character (n) varying	$1 \leq n \leq 254$	varchar	1 to 254 char
binary fixed (n)	$1 \leq n \leq 15$	smallint (2-byte)	-32,768 to +32,767
binary fixed (n)	$16 \leq n \leq 31$	integer(4) (4-byte)	-2,147,483,648 to +2,147,483,647
binary float(p)	$1 \leq n \leq 21$	float4 (4-byte)	-1.0e+38 to +1.0e+38 (7-digit precision)
decimal float (n)	$1 \leq n \leq 6$		
binary float (n)	$22 \leq n \leq 53$	float (8-byte)	+1.0e+38 (16-digit precision)
decimal float (p)	$7 \leq n \leq 16$		
fixed decimal (7)	-9,999,999 to +9,999,999	float4 (4-byte)	-1.0e+38 to +1.0e+38 (7-digit precision)
fixed decimal (15)	-999,999,999,999,999 to +999,999,999,999,999	float (8-byte)	-1.0e+38 to +1.0e+38 (16-digit precision)

The following table shows the 370 Assembler and EDBC data types:

370 Assembler Data Type	EDBC Data Type	EDBC Range
C = alphanumeric	char	1 to 2000 char
	varchar	1 to 2000 char
H = binary halfword	smallint (2-byte)	-32,768 to +32,767
F = binary fullword	integer(4) (4-byte)	-2,147,483,648 to +2,147,483,647
X = hexadecimal	char	1 to 2000 char
	varchar	1 to 2000 char
E = float	float4 (4-byte)	-1.0e+38 to +1.0e+38 (7-digit precision)
D = float doubleword	float (8-byte)	-1.0e+38 to +1.0e+38 (15-digit precision)

370 Assembler Data Type	EDBC Data Type	EDBC Range
P = packed decimal	float4 (4-byte)	-1.0e±38 to +1.0e±38 (7-digit precision)
	float (8-byte)	-1.0e±38 to +1.0e±38 (16-digit precision)
	integer (4-byte) (maximum of 9 digits)	-2,147,483,648 to +2,147,483,647

EDBC Data Types

The following table lists the full range of EDBC data types, and shows, which are supported by the Vantage gateway. The range is specified for the data types the gateway supports.

If the Vantage data type corresponds precisely to the EDBC data type, then each Vantage record will match each EDBC row. In most cases, some degree of translation is required by the gateway. This information is specified in the register table external formats.

Note also that the Vantage EWS interface module is aware of some Vantage data formats that do not map directly to EDBC formats, and that this module will internally convert some Vantage formatted data into EDBC compatible formats in real-time. Therefore, be careful when you create your own registration scripts by referring to the scripts generated by the VANTEDBC utility. This utility is aware of these internal conversions, and creates registration scripts with an understanding of what conversions will take place within the EWS module. It is recommended that you take scripts generated by the VANTEDBC utility and modify those scripts as needed.

The following table shows the Vantage gateway support for EDBC data types:

EDBC Data Type	Use in Vantage Gateway Environment
char	Fixed length string of 1 to 2000 characters
c	Not supported
varchar	Variable length string of 1 to 2000 characters
vchar	Not supported
text	Variable length string of 1 to 2000 characters
integer1	One-byte integer (same as I1)
integer2	Two-byte integer (same as I2 or smallint)
integer4	Four-byte integer (same as I4 or integer)

EDBC Data Type	Use in Vantage Gateway Environment
smallint	Exact numeric 2-byte integer. Range: -32,768 to +32,767
integer	Exact numeric 4-byte integer. Range: -2,147,483,648 to +2,147,483,647
float4	Floating point 4-byte. Range: -1.0e±79 to +1.0e±79 (7-digit precision)
float float8	Floating point 8-byte. Range: -1.0e±79 to +1.0e±79 (16-digit precision)
date	12-byte internal EDBC OpenSQL format or 4-byte hex with numerous formats
money	Floating point 8-byte cents. Range: -99,999,999,999.99 to +99,999,999,999.99
table_key	Not supported
object_key	Not supported

EDBC Character Strings

A character string is a sequence of characters. Upper and lowercase characters within a string are accepted literally. Char strings may contain any character, including non-printing and null characters. Varchar strings can contain any character, including non-printing and null characters.

SQL recognizes blanks in character string comparisons. For example, the character string:

the house is around the corner

is treated differently than:

thehouseisaroundthecorner

In comparing strings of unequal length, SQL adds blanks to the end of the shorter string to equal the length of the longer string.

Char Data Type

This data type specifies a fixed-length character data type. The syntax for this data type is:

char(*len*) [**is** '*external_format*']

where:

<i>len</i>	Specifies the total length of the column. The corresponding Vantage field is the same length. The maximum value for this parameter is 2000 bytes.
<i>external_format</i>	Specifies the format of the Vantage field in the Vantage record. You may specify HEXPRINT as the value. HEXPRINT directs EDBC to send the data back in displayable hexadecimal format. Each byte of data requires two characters for display; therefore, the length of the char field must be a multiple of two.

Varchar Data Type

The varchar data type specifies a variable-length character data type. It is maintained internally by EDBC as a 2-byte length field followed by the character data, which is padded with filler up to the length specified in the table registration. The 2-byte length field contains the actual length of the data entered. The user does not see the length field. The syntax of this data type is:

varchar(*len*) [**is** '*external_format*']

where:

<i>len</i>	Specifies the total character length of the column. The corresponding character portion of the Vantage field is the same length. The maximum value for this parameter is 2000 bytes.
<i>external_format</i>	Specifies the format of the Vantage field in the Vantage record. You may specify one of the following options: <ul style="list-style-type: none"> ■ VCHAR 1 Specifies the length of the length field as one byte. This allows a maximum character length of 255 bytes. ■ VCHAR 2 Specifies the length of the length field as two bytes. This is the default. ■ VCHAR 4 Specifies the length of the length field as four bytes. UCASE is an optional parameter. UCASE specifies that the field is stored and maintained in uppercase. The gateway converts any lowercase input to uppercase before storing the data in the gateway.

The following are examples of the varchar data type:

`fld1 varchar(200)`

This first example defines a field in both the gateway and Vantage that consists of a 2-byte length field followed by 200 bytes of character data.

`fld2 varchar(200) is 'vchar 1 ucase'`

This example defines a Vantage field that consists of a 1-byte length field followed by 200 bytes of character data that is converted to uppercase by the gateway before presenting the data to Vantage.

EDBC Numeric Data Types

The exact numeric data types are smallint (2-byte) and integer (4-byte). Integers that exceed the permissible range are converted to floating point. Integers can contain no fractional part.

The approximate numeric data types are float4 (4-byte) and float (8-byte). (Float is synonymous for float8.) An exponent of one (1) is assumed if none is specified. A float4 data type offers seven digits of precision. A float data type offers 16 digits of precision, stored as eight bytes. By default, the period character (.) is used to indicate the decimal point.

In addition to these EDBC numeric data types, Vantage objects may contain packed and zoned decimal data. The gateway will convert this data to either integer or floating point format for the EDBC application. The syntax for defining the decimal data is:

col_name format is 'decimal(x,y)'

where:

<i>col_name</i>	Indicates the field in the table.
<i>format</i>	Indicates a valid EDBC data type.
<i>decimal</i>	Indicates whether the external data is in packed or zoned decimal format. Use <code>decimal</code> or <code>packed_decimal</code> for packed and <code>zoned_decimal</code> for zoned.
<i>x</i>	Indicates the total number of digits, including the decimal fraction, to be converted. Note that in order to preserve data accuracy, you can register decimal data where <i>x</i> is greater than 15 only in read-only partitions.
<i>y</i>	Indicates the number of digits following the decimal point. This is also referred to as the precision, or scale.

For example, the following shows how the PART file is mapped using the decimal clause to convert a packed value in column “price” to a float8 data type:

```
register table part
    (part_num char(3),
     material char(10),
     price float8 is 'decimal(5,2)')
...

```

In this example, the definition for “price” would correspond to the COBOL picture clause S9(3)V99 COMP-3.

The float data type cannot be used to perform exact matches. Therefore, if an application requires exact matches during query qualification, define the Vantage decimal data as an integer. However, if you use a float format, you must coerce the float field into integer format.

For example, if you have a packed decimal field defined as decimal(5,0) and it must be used in an exact match qualification, register the column this way:

```
register table xyz
    decimal_data float8 is 'decimal(5,0)'

```

In order to access the decimal_data field as an integer, you must create a view on table xyz that maps the field to an integer value:

```
create view view_xyz
    (...
     decimal_data,
     ...)
as
    select (...
           integer(xyz.decimal_data + 0.5),
           ...)
    from xyz;

```

The limitation of this is that the field length is limited to the maximum size of the integer field.

Smallint Data Type

The smallint data type specifies a two-byte integer. The Vantage field must be a one or two-byte integer. The syntax of this data type is:

```
smallint [is 'external_format']
```

External Format	Description	Parameters
Integer (<i>width</i>)	Specifies that the Vantage width field is an OS/390 integer.	Number of decimal digits defined in the field. The maximum value is 2.

Integer Data Type

The integer data type specifies a four-byte integer. The Vantage field can be an OS/390 integer, packed decimal, or zoned decimal. The syntax of this data type is:

integer [is 'external_format']

The following table describes the *external_format* clause:

External Format	Description	Parameters
decimal (<i>width, precision</i>) or packed_decimal (<i>width, precision</i>)	Specifies that the Vantage field is an OS/390 packed decimal field.	<p>The following parameters can be specified:</p> <ul style="list-style-type: none"> ■ <i>width</i> Number of decimal digits defined in the field. The maximum value is 9. The field size is width/2 + 1 bytes. ■ <i>precision</i> Number of places to the right of the decimal point.
zoned_decimal (<i>width, precision</i>)	Specifies that the Vantage field is an OS/390 zoned decimal field.	<p>The following parameters can be specified:</p> <ul style="list-style-type: none"> ■ <i>width</i> Number of decimal digits defined in the field. The maximum value is 9. The field size is width bytes. ■ <i>precision</i> Number of places to the right of the decimal point.
integer (<i>width</i>)	Specifies that the Vantage <i>width</i> field is an OS/390 integer.	Number of decimal digits defined in the field. The maximum value is 4.

The gateway performs a validity check on the contents of each field in the Vantage record before it permits the contents of the record to be modified. This feature preserves the integrity of the Vantage data in the event of a mapping error by the database administrator.

Float, Float8, and Float4 Data Types

The float, float8, and float4 data types specify a floating point quantity. The possible OS/390 Vantage fields that correspond to these data types are:

- float long
- float short
- packed decimal
- zoned decimal

The EDBC column widths that correspond to these data types are:

float	8
float8	8
float4	4

You can specify any of the EDBC float data types, with or without an external format. The syntax is:

- float** [**is** '*external_format*']
- float4** [**is** '*external_format*']
- float8** [**is** '*external_format*']

If no external format is specified, then the data types are mapped to the OS/390 data types as shown in the following table:

Data Type	OS/390 Data Type	Vantage Width
float	OS/390 float long	8
float8	OS/390 float long	8
float4	OS/390 float short	4

If you specify an external format, the data types are mapped as shown in this table:

External Format	Description	Parameters
decimal (<i>width, precision</i>) or packed_decimal (<i>width, precision</i>)	Specifies that the Vantage field is an OS/390 packed decimal field.	<p>The following parameters can be specified.</p> <ul style="list-style-type: none"> ■ width Number of decimal digits defined in the field. The maximum value is 7 for float4, and 15 for float and float8. The field size is $\text{width}/2 + 1$ bytes. ■ precision Number of places to the right of the decimal point.
zoned_decimal (<i>width, precision</i>)	Specifies that the Vantage field is an OS/390 zoned decimal field.	<p>The following parameters can be specified.</p> <ul style="list-style-type: none"> ■ <i>width</i> Number of decimal digits defined in the field. The maximum value is 7 for float4 and 15 for float and float8. The field size is <i>width</i> bytes. ■ <i>precision</i> Number of places to the right of the decimal point.

The gateway does a validity check on the contents of each field in the Vantage record before it permits the contents of the record to be modified. This feature preserves the integrity of the Vantage data in the event of a mapping error by the database administrator.

Date Data Type

The gateway supports a multitude of date formats including the ODBC OpenSQL date data type. For a list of input and output formats for the date data type, refer to the *ODBC OpenSQL Reference Guide*.

The syntax for the date data type is as follows:

col_name date [is '*external_format*']

where *external_format* specifies one of the following date formats:

YYDDMM	YY0DDD
YYMM	MMYY
YYDDD	MMDDYY
YYYYDDMM	YYYYMMDD
YYYYMM	MMYYYY
YYYYDDD	MMDDYYYY
0CYDDDDF	CYYMMDDF
YYMMDD	

where the format is defined as shown in the following table:

Date	Definition
YY	Low order 2 digits of year
MM	Month of the year
DD	Day of the month
DDD	Julian day of the year
YYYY	4 digits of year
0CYDDDDF	Date returned from the OS/390 TIME SVC (fullword packed field) where C = century 0 = 1900 1 = 2000 2 = 2100
CYYMMDDF	Packed decimal field where C = century 0 = 1900 1 = 2000 2 = 2100

The date specified by the `external_format` is assumed to be store as a OS/390 fullword binary value (i4). If it is not, an overriding internal format clause must be specified. The syntax for specifying an overriding value is:

col_name date is 'external_format decimal(x,y)'

where:

<i>col_name</i>	Indicates the column name in the table.
<i>date</i>	Indicates that this is a date field.
<i>external_format</i>	DATE format (for example, YYYYDDMM)
<i>type</i>	Indicates the internal format of the date in the Vantage record. The acceptable values are: <ul style="list-style-type: none"> ■ decimal or packed_decimal ■ zoned_decimal
<i>unsigned</i>	Indicates that decimal or packed_decimal is an unsigned value. The default is signed. Not valid for zoned_decimal.
<i>x</i>	Indicates the length of the date field in the Vantage record.
<i>y</i>	Indicates the precision (number of places right of the decimal point). Must be specified as 0.

The following are examples of the date data type statement:

Date Data Type Statement	Description
<code>invoice_date date is 'MMDDYY'</code>	Date is fullword binary.
<code>sales_date date is 'YYYYMMDD zoned_decimal (8,0)'</code>	Date is zoned decimal.
<code>paydate date is 'YYYYDDD unsigned packed_decimal (7,0)'</code>	Date is an unsigned decimal value.
<code>birth_day date is 'YYYYDDMM' packed_decimal</code>	Date is a signed packed decimal value.

If you do not specify an external format with the DATE data type (for example, `sales_date date`), then the Vantage date field is assumed to be 12 bytes in length and in the EDBC OpenSQL date data type format as described in the following table.

The following table shows the description of date data type:

Offset	Type	Len	Description
0	bitstring	1	Status bits: <ul style="list-style-type: none"> ■ 0x00 - Date is the empty date ■ 0x01 - Absolute point in time ■ 0x02 - Date is a time interval ■ 0x04 - Year specified ■ 0x08 - Month specified ■ 0x10 - Day specified ■ 0x20 - Time specified
1	bitstring	1	High order bits of day
2	binary	2	Year (number years for intervals)
4	binary	4	Month (number months of intervals)
4	binary	2	Day (number days for intervals) Refer also to highday for the high order bits of the day, if more than will fit into halfword.
8	binary	4	Time in milliseconds from 00:00

One adVantage of using the EDBC OpenSQL format with no external format is that time and century are also maintained.

Dates are converted based on the external format specifications. They are converted to or from the EDBC OpenSQL date format, depending on the processing being performed. When a query is issued against a Vantage object, date fields are converted from the external format to the EDBC OpenSQL date format (described above). When a client inserts into or updates a Vantage object, existing date fields are converted from the EDBC OpenSQL date format to the format specified by the external format specifications.

Caution! When dates are converted from a Vantage date to an EDBC OpenSQL date format, and when an external_format specification containing a 2 digit year is encountered, the II_DATE_CENTURY_BOUNDARY logical is applied in an attempt to determine the correct century. If the II_DATE_CENTURY_BOUNDARY is not specified, the current century is assumed. This is a best guess attempt and may not always be correct. To be completely accurate, users must insure that their Vantage data is Y2K compliant prior to accessing it via the Vantage gateway.

The following table depicts various external formats and displays the internal hexadecimal representation of the date:

External_format Specifications	Date	Internal Representation
YYMMDD	990207	0F1C01
YYMMDD zoned_decimal (6,0)	990207	F9F9F0F2F0F7
YYMMDD unsigned packed_decimal (4,0)	990207	0990207F
YYYYDDMM	19990702	013108AE
YYYYDDMM packed_decimal (5,0)	19990702	019990702C
YYYYDDMM zoned_decimal (8,0)	199990702	F1F9F9F9F0F7F0F2
0CYYDDDF	1999038	0099038F

Money Data Type

Money fields are internally stored as float8 fields. The values in these fields are the number of cents in the money field. The Vantage gateway uses the same format and thus expects the Vantage field to be in float8 cents format. No external formats (such as decimal) are currently supported for money. If external formats are required, use the float8 data type instead.

The external (display) range for the money data type is:

maximum = 99,999,999,999,999.99
 minimum = -99,999,999,999,999.99

This means that the maximum and minimum internal values for the money field are:

maximum = 9,999,999,999,999,999.0
 minimum = -9,999,999,999,999,999.0

Note that the internal money value is represented in cents.

Source of the Gateway Object

The syntax used to specify the **from** 'source' clause must be strictly followed to properly identify Vantage objects to EDBC.

Accessing an Existing Vantage Object

To access a defined Vantage object, use this syntax to specify the location of the Vantage cluster:

```
from 'Vantage_object_name PARM=VIOO'
```

The following table describes the parameters of the from clause (Existing Objects):

Parameter	Description
Vantage_object_name	This value must be a valid, Vantage object name, and must be 9-characters long padded to the right with blanks.
PARM=VIOO	This is a required value within the <i>from</i> clause and tells EDBC to call the Vantage interface routine to process requests directed to this object.

Access Methods and EDBC Structures

You must specify the type of EDBC structure that corresponds to the access method for the Vantage object using the structure clause of the register table statement. As the Vantage implementation emulates VSAM KSDS, you should always use the “sortkeyed” structure for Vantage objects.

Performance Impact

You can significantly improve SQL statement performance when using sortkeyed objects if you specify key fields in the where clause. If the object has no key (such as for a VSAM ESDS) or if no key is qualified in the SQL statement, then a full table scan is performed. Queries will be optimized by doing partial scans whenever leading (leftmost) key fields are specified in the where clause for sortkeyed files.

If you issue an SQL statement that qualifies non-key fields, a full scan of the table will be done. For example, if you issue the following statement against a fictional sample vehicle table, a full scan of the table will occur if mileage is not a key column:

```
select * from vehicle
```

```
where mileage > 30.00  
and mileage < 100.00;
```

To prevent a full table scan, create the equivalent of a Vantage alternate index specifying an index on the mileage field by creating a new EDBC table specifying the mileage field as a key instead of the VANTEDBC-generated default key field.

A partial-key search partially defines the search criteria on a key column. If you issue a partial-key search, a full table scan is done if the leading (leftmost) portion of the key is not specified. For example, this query requires a full table scan to locate and compare data:

```
select * from vehicle  
where vehicle_id like '_4%';
```

If you qualify the leading part of the key, the search criteria narrows and a full table scan is not performed:

```
select * from vehicle  
where vehicle_id like 'v4%';
```

Rows

Use the rows parameter to specify the number of rows in the table as closely as possible. The default number of rows for a registered table is 1000. When an EDBC user queries the table, the gateway data manager uses this value to plan the most efficient way to execute the query.

For dynamically defined objects, the value assigned to the rows clause is used to specify the space allocation requirements. The initial space allocation required when the gateway creates the Vantage object is based on the record size multiplied by the number of rows specified. If there is insufficient space on the disk when the register statement is processed, the statement will fail.

Repeating Groups

The following sections describe how repeating groups are implemented in the EDBC gateway for Vantage:

- Repeating group restrictions
- Registering a Vantage object with a repeating group as an EDBC table

- Retrieving data from a table with a repeating group
- Special considerations when working with repeating groups

Repeating Group Restrictions

Support for repeating groups in the EDBC for Vantage gateway is provided with the following restrictions:

- UPDATE access to tables mapping a Vantage record consisting of one or more repeating groups is not supported.
- Only one repeating group per table is supported: Vantage records composed of multiple repeating groups are supported as long as every group (except the last one) consists of a fixed number of occurrences. Each repeated group can be registered as a separate table.
- The repeating group must be located at the end of the mapped section of the record. Mapping of data following a repeating group is not supported.
- Nested repeating groups are not supported unless inner groups are de-normalized at registration time.
- Variable-length elements are not supported in repeating groups.
- The last field in a repeating group must be defined (even if it is a FILLER).

Registering a Table with a Repeating Group

The following example shows how to register a Vantage file that contains a repeating group:

```
REGISTER TABLE employee (
  last_name      char(20),
  first_name     char(20),
  gender         char,
  age            smallint,
  salary         integer,
  sd_total       integer, /* count of spouse + dependents */
  sd_occurrence  integer is 'occurs(sd_total)',
  sd_last_name   char(20), /* may have different last name */
  sd_first_name  char(20),
  status         char, /* S: spouse, D: dependent */
  sd_gender      char,
  sd_age         smallint)
AS IMPORT FROM '..... PARM=VI00' WITH DBMS=VSAM, STRUCTURE=SORTKEYED,
KEY=(last_name), ROWS=nnnnnn, NODUPLICATES.
```

The Occurs Data Type

The presence of a repeating group is indicated to EDBC during the registration process by the existence of a column with a numeric data type and with an external format of "occurs". This column must be defined immediately before the first column in the repeating group. The "occurs" external format indicates that all the columns following it map to fields in a repeating group.

There are two forms of the occurs external format clause:

- Occurs(nn) indicates a fixed number of occurrences of a repeating group.
- Occurs(column_name) where column_name is a column residing in the fixed portion of the Vantage record that indicates the number of occurrences of the group in the current record.

The "occurs" column is a "virtual" column (it is a computed column that does not physically exist in the underlying Vantage record). The value contained in this column is the number of the element (0 if there are no elements, 1 for the first one, 2 for the second one) in the repeating group. Only one "occurs" column per table is allowed.

Retrieving Data from a Repeating Group

The following SQL statement retrieves repeating group data:

```
SELECT * FROM employee;
```

This statement breaks each Vantage record into sd_total number of rows. Each row consists of the fixed portion of the Vantage record, the "occurs" column and a single element of the group.

Special Considerations When Working with Repeating Groups

If the computed value of the "occurs" field is zero, EDBC returns a single row with every column in the repeating group section of the row set to binary zeroes.

For tables mapping repeating groups, the ROWS parameter of the REGISTER TABLE statement should be set to the number of records in the Vantage object times the average number of elements in the repeating group. When the repeating group has a fixed number of elements, the number or ROWS should be set to the number of records in the file times the number of elements in the repeating group.

Tables mapping repeating groups must specify "duplicates" whenever keys are specified.

VALUE Clause

The VALUE parameter of the IS clause is used to tell EDBC to filter records based on a constant value at a fixed offset within each record. The VALUE is specified as a string of hexadecimal digits within parentheses. This string must be specified as an exact match for the data as it resides in the file for a specific record to be selected.

Multiple columns with a VALUE external data type are supported in a single table, but each column cannot exceed 4 bytes (8 hexadecimal digits) in length. If a VALUE field is longer than 4 bytes, it is necessary to break it up into multiple columns where each column is no longer than 4 bytes. VALUE filtering is performed after the data is returned by Vantage. The only situation where VALUE filtering is more efficient than a WHERE clause is when it is applied to non-key fields. Key fields should always be specified in WHERE clauses for ultimate performance. This is particularly critical when working with large files or joining multiple files of any size. Specifying columns with a VALUE external attribute in a WHERE clause is allowed.

Registering a Table with a VALUE Clause

Here is an example of a COBOL copybook with a VALUE clause—fields PTIQIND-CD1 and PTIQINQ-CD2 indicate the record type corresponding to this copybook:

```

000700 01 PTIQINQ-SEGMENT.                                00070005
000800 05 PTIQINQ-CUSTOMER-NUMBER                      PIC 9(9) COMP.    00071007
000800 05 PTIQINQ-CUST-EFF-9JDATE                      PIC 9(7) .        00072007
001100 05 FILLER                                        PIC X(14) .       00073007
000800 05 PTIQINQ-SEGMENT-LENGTH                      PIC S9(4) COMP.  00080005
000900 05 PTIQINQ-NUM-ENTRIES                          PIC 9(3) .        00090005
001000 05 FILLER                                        PIC X(16) .       00100001
001100 05 PTIQINQ-INT-CD.                               00110005
001200 10 PTIQINQ-CD1                                 PIC 9(02) VALUE 1. 00120005
001300 10 PTIQINQ-CD2                                 PIC 9(02) VALUE 4. 00130005
001400 05 PTIQINQ-ENTRY OCCURS 1 TO 100 TIMES         00140005
001500          DEPENDENT ON PTIQINQ-NUM-ENTRIES.      00150005
001600 10 PTIQINQ-CONTACT-DATE                       PIC 9(7) COMP-3.  00160005
001700 10 PTIQINQ-COMMENT                             PIC X(16) .       00170005

```

The following REGISTER table script was generated by the COBOL Data Mapper:

```
register table ptiqinq(
  ptiqinq_customer_number integer is 'offset(0)',
  ptiqinq_cust_eff_9jdate char(7) is 'offset(4)',
  filler1 char(14) is 'offset(11)',
  ptiqinq_segment_length smallint is 'offset(25)',
  ptiqinq_num_entries integer is 'offset(27) zoned_decimal(3,0)',
  filler2 char(16) is 'offset(30)',
  ptiqinq_cd1 char(2) is 'offset(46) value(f0f1)',
  ptiqinq_cd2 char(2) is 'offset(48) value(f0f4)',
  ptiqinq_entry integer is 'offset(50) occurs(ptiqinq_num_entries)',
  ptiqinq_contact_date integer is 'offset(0) decimal(7,0)',
  ptiqinq_comment char(16) is 'offset(4)'
)
as import from '.... PARM=VI00'
with
  dbms = VSAM,
  structure = sortkeyed,
  key = (ptiqinq_customer_number),
  rows = 500;\p\g
```

Mapping Redefined Storage

Multiple record layouts can occupy the same storage. Redefining record layouts allows an application to equate two or more different names to the same storage area. In addition, one of the declared names may represent all or part of the same storage that is assigned to the second name. This is a useful feature in programming languages such as PL/I.

The following PL/I statements illustrate this concept:

```
DECLARE NAME CHAR(20) INITIAL('JAMES WILEY RHOADES');
DECLARE FIRST CHAR(6) DEFINED NAME;
DECLARE FIRST_MIDDLE CHAR(12) DEFINED NAME;
```

In this example, NAME is the base identifier. This is the variable name to which other variable names are equated or defined. The base identifier must be at least equal to or greater in length than the other variables that are overlay-defined on it. There may be more than one variable name that is overlay-defined on a base identifier. In this example, FIRST and FIRST_MIDDLE are the variable names that are overlay defined on the NAME variable.

NAME, FIRST, and FIRST_MIDDLE occupy the same storage area. FIRST is assigned to the six leftmost characters. FIRST_MIDDLE is a second variable name that is overlay-defined on NAME. It occupies the 12 leftmost characters.

The Vantage gateway supports mapping of the base identifier and overlay-defined variables. One method is to map the overlay-defined items as a relational table and map the base identifier as another relational table. The is *'offset()'* clause is an alternative method.

Using the example above, the following register table syntax maps the overlay-defined items:

```
register table table_name
    (key char(3),
     first char(6),
     middle char(6),
     last char(8))
as import
from 'example.dataset'
...
```

The following register table syntax maps the base identifier as a relational table:

```
register table table_name2
    (key char(3),
     name char(20))
as import
from 'example.dataset'
...
```

Use the following statement to map the base identifier as a view:

```
create view view_name
    (name char(20))
as select * from table_name;
```

When a row is overlay-defined based upon a variable data type, each overlay-definition is mapped as a relational table. The following example illustrates how different overlay-defined rows are mapped when a variable is used to determine what view the application uses.

```
register table table1
    (key char(3),
     type integer,
     employee char(10),
     salary float)
as import from 'example.dataset'
...

register table table2
    (key char(3),
     type integer,
     first_name char(6),
     middle_name char(4),
     salary float)
as import from 'example.dataset'
...
```

In the above example, table1 contains a single field called employee that is the name of an employee. In table2, the name of an employee is contained in two fields, first_name and middle_name. Both tables contain identical data.

Grants of Permission

The gateway has a built-in hierarchical security system in which the DBA for a database controls user access to the database and tables it contains. Data access is permitted by grants of permissions on tables and indexes.

In the Vantage environment, the gateway DBA owns all data registered through the gateway. Once a Vantage object is registered, the gateway enforces the permissions assigned for that table or index. These constraints on access to relational tables and indexes operate in addition to the restrictions imposed by security facilities like CA-ACF2, RACF, or CA-TSS to which the gateway can be interfaced.

The SQL grant statement has the general form:

grant *privilege on object to whom*

- *Privilege* specifies the type of permissions being granted.
- *Object* names the table or index on which privileges are being granted.
- *Whom* names the user (or group of users, or public) to whom authorization is being granted.

For a more complete description of the grant statement, refer to the *EDBC OpenSQL Reference Guide*.

Individual User Authorizations

Privileges defined by a particular grant statement may be issued to one or more users, as specified with Net user IDs. You can also grant to *public*, which grants the specified permissions to all user IDs.

Granting a privilege allows the user to issue the specified type of query on the table or index. These are enabling permissions. If no permission is given, the default is to prohibit access. The *select* and *all* query privileges can be granted on registered Vantage objects.

Sharing Vantage Objects Among EDBC Users

The Vantage gateway allows access of Vantage data concurrent with other Vantage gateway users and other Vantage applications.

Logging and recovery is not supported for the local Vantage access.

The gateway partition contains all catalog information that is used to access Vantage data and support EDBC applications and utilities. This partition is contained within the gateway user area, which is a VSAM Linear Data Set (LDS). The remote user's OS/390 authorization identifier is used to select the Vantage gateway user area that is associated with that particular user.

Utilities allow the gateway DBA to easily add new users or to clone a copy of an existing gateway user area for a new user.

Building Register Table Scripts

The gateway provides a utility interface to assist a gateway DBA in registering Vantage objects as relational tables. This utility interface is the Vantage EDBC Data Mapper (VANTEDBC).

The Vantage EDBC Data Mapper generates register table statements from live Vantage subsystems. The mapper runs on any mainframe with TCP/IP access to an active Vantage subsystem. The output from the mapper includes JCL that may be used to execute each of the registration scripts for each Vantage object discovered by the utility (PDS member \$RUNJOB).

The Vantage EDBC Data Mapper utility should always be run as a batch job, and not under TSO because it will prompt you to enter a password it should use in connecting with the target Vantage system through the WTOR facility. Model JCL is shipped with your product, which you should customize for your environment.

The Vantage EDBC Data Mapper utility requires that you identify a target PDS into which the utility will place the registration scripts for the target Vantage subsystem.

Note: This utility will overwrite members in the target PDS as necessary. It also requires that you identify an output DD (REPORT) it may use to generate a report on the run of the utility. You must also identify the target Vantage subsystem and a security user ID using execution parms.

Here is an example of the model JCL for the Vantage EDBC Data Mapper:

```
// (valid job card)
//BUILD EXEC PGM=VANTEDBC,
// PARM='tcpip-name,xxx.xxx.xxx.xxx:pppp,user id,obj'
//SCRIPTS DD DISP=SHR,DSN=EDBC.Vantage.SCRIPTxx
//REPORT DD SYSOUT=*
```

If you are going to be accessing multiple Vantage subsystems, make sure that you point the SCRIPTS DD to a unique PDS for each subsystem.

The format of the PARM data on the EXEC card is as follows. The first three parameters are all required parameters:

- First parameter – The name of your TCP/IP address space for the OS/390 system on which you plan to run the Vantage EDBC Data Mapper utility.
- Second parameter – The TCP/IP address and port number of the target Vantage subsystem. The target Vantage subsystem does not have to be running on the same system as the Vantage EDBC Data Mapper.
- Third parameter – The OS/390 user ID to be used in connecting to the target Vantage subsystem; (you will be prompted through a WTOR for the password).
- Fourth parameter (optional) – A specific Vantage object name if specified. This allows you to run the utility to regenerate the script for a specific Vantage object instead of the normal mode where all objects are scripted.

To generate registration scripts for all Vantage objects, do not specify a fourth parameter on the EXEC card.

Testing Access to a Registered Table

You can use the Terminal Monitor to verify local access to the tables. This process provides limited information about the tables, but does allow you to confirm that the tables exist and are properly formatted.

To verify local access:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section in the “Using the IGWFDDBA Utilities” chapter.
2. Enter **2.5** in the Selection field and press Enter to invoke the Gateway Using Terminal Monitor panel.
3. Check the name of the partition and the server class. The server class value must be VSAM. Correct these values if necessary. For example:

```
iivpvm/VSAM
```

4. Press Enter to accept the contents of the screen.

If you allocated the parms correctly, the screen displays the Terminal Monitor banner, then the * prompt.

5. Use the help command to test access to the Terminal Monitor. Use one of the following methods:

- a. Type:

```
help\g
```

then press Enter.

- b. Type:
help *tablename*\g
then press Enter.
 - c. Type:
select * from *tablename*\g
then press Enter.
6. To exit from the Terminal Monitor, type \q, then press Enter.
 7. Return to the Vantage gateway menu by pressing PF3.

Troubleshooting Registration Problems

If Vantage data is improperly defined to the gateway, unpredictable results can occur. For example, arithmetic can be performed on character fields, and integers can be taken as floating point numbers. If you encounter unusual results, consider the following strategies:

- From the SQL Terminal Monitor help command, lay out the table rows. Compare this row layout with several records dumped from the data set. Ensure that they match.
- Check the register table command that defined the table, or the output from the EDBC Gateway Facility DBA (IGWFDBA) register table utilities, to make sure that the data conversions properly reflect the actual data in the underlying Vantage objects by referring to scripts generated by the Vantage EDBC Data Mapper (VANTEDBC).
- Make sure that any index definitions are correct. The key fields and their order must match those in the actual underlying index.
- The GET and GWGET traces are useful tools for debugging mapping errors. See the Debugging the Gateway section in the “Maintaining the Gateway” chapter for more information.

When you identify the problem, remove the improper template and reregister the gateway table correctly.

Removing a Gateway Table or Index

To remove the definition of a Vantage object from a gateway partition, use the SQL `remove table` command.

For predefined Vantage objects, the `remove table` command deletes all standard catalog entries for a registered table or index, including any related views or permits. The underlying Vantage object is not affected.

The `remove table` command has the syntax:

remove table *tablename*

where *tablename* is the name of the relational table or index that corresponds to a registered Vantage object.

To issue the `remove table` command, go into the Terminal Monitor using the following procedure:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section in the “Using the IGWFDDBA Utilities” chapter.
2. Enter **2.5** in the Selection field and press Enter to invoke the gateway Using Terminal Monitor panel.
3. If necessary, type the name of the partition from which you want to remove the registered table or index.

The server class must always be VSAM.

4. Press Enter to accept the contents of the screen.

If you allocated the parameters correctly, the screen displays the Terminal Monitor banner, then the * prompt.

5. Issue the SQL `remove table` command, by typing:

remove table *tablename*

6. To exit from the Terminal Monitor, type `\q`, then press Enter.
7. To return to the Vantage gateway menu, press PF3.

Note: Do not use the `drop table` command to remove a Vantage object from a gateway partition. The `drop table` command removes the catalog entries for the Vantage object, but does not delete it from the gateway partition.

Maintaining the Gateway

This chapter describes a series of maintenance and troubleshooting procedures that are carried out by the gateway DBA. Most of these utilities are carried out using the IGWFDBA utilities.

This chapter explains how to:

- Start and stop the EDBC server
- Maintain the EDBC server address space
- Run DBMS utilities that perform an array of functions on gateway areas and partitions
- Define a second EDBC server

For more information on using the IGWFDBA utilities, see the “Using the IGWFDBA Utilities” chapter.

Starting and Stopping the EDBC Server

Use these procedures to start and stop the EDBC server.

Starting the EDBC Server

When you start up the EDBC server, you initialize the:

- Gateway name server
- Gateway operator interface
- Gateway communications server
- SNA LU0, SNA LU62, IBM TCP/IP, KNET TCP/IP, CCI, or Interlink SNS/TCP protocol server

To start the EDBC server address space using the default installation value for the subsystem, the OS/390 operator issues the following command:

S procname.identifier

For example, with the default installation value for the identifier, this command is:

```
S EDBC11.EDBC
```

The following messages appear on the OS/390 console:

```
EDBC: EDBC MVS Server Default Version for CPU id xx-xxxxxx-xxxx
EDBC: Initializing Operator Interface Subtask
EDBC: Initializing Housekeeping Subtask
EDBC: Initializing communication queue servers
EDBC: Initializing communication drivers
EDBC: SNA_LU0 Protocol Server Initialization Begun
EDBC: SNA_LU0 Opened ACB IIS1GWS1
EDBC: SNA_LU0 Protocol Server Initialization Complete
EDBC: TCP_IBM Protocol Server Initialization Begun
EDBC: TCP_IBM Listening on Port 2610
EDBC: TCP_IBM Protocol Server Initialization Complete
EDBC: TCP_SNS protocol server Initialization Begun
EDBC: VSAM connection ACTIVated
EDBC: EDBC Server Initialization Complete
```

Note: This example shows two of the protocol servers initializing. You may have other protocol servers installed at your site.

After initialization is successfully completed, the EDBC server is ready to process requests.

Stopping the EDBC Server

Closing down the EDBC server terminates the activity of all the gateways. To shut down the EDBC server, use the following OS/390 operator command:

```
P identifier
```

For example, with the default installation value for the identifier, this command is:

```
P EDBC
```

This command causes the EDBC server to carry out termination processing. After you issue this command, the following messages appear on the OS/390 console:

```
EDBC: EDBC Server shutdown in progress
EDBC: Wait for Operator Interface to end
EDBC: Operator Interface ended
EDBC: SNA_LU0 Protocol Server Termination Begun
EDBC: SNA_LU0 Protocol Server Termination Complete
EDBC: TCP_IBM Protocol Server Termination Begun
EDBC: TCP_IBM Protocol Server Termination Complete
EDBC: Housekeeping shutdown complete
EDBC: Terminating communication queue servers
EDBC: EDBC Server shutdown complete (rc=0)
```

Bringing Up the EDBC Server as a Batch Job

To start up the EDBC server as a batch job, do the following:

1. Use either one of the following members from the SAMPLE.CNTL library to start the EDBC server as a batch job:
 - xxxxEDBC where xxxx is the OS/390 subsystem ID.
xxxxEDBC contains all the JCL for the server.
 - STARTyy where yy is the gateway installation ID.
STARTyy executes a procedure to start the server.
2. Submit the job for execution.

The console display the series of messages shown in Step 1, terminating in the message:

```
EDBC: EDBC server initialization complete
```

Maintaining the EDBC Server Address Space

You can use the OS/390 F (MODIFY) command to carry out a series of operations for the EDBC server address space. From the OS/390 operator console you can:

- Display active users and internal service tasks, with information about the status of each thread
- Activate the specified protocol server
- Inactivate the specified protocol server, user ID, or user context
- Set tracing to debug the specified protocol server
- Turn off tracing
- Dump the entire address space for troubleshooting
- Get help for this utility

The commands are all variations on the following format:

F *identifier, command*

A blank or a comma is a valid delimiter. Extraneous blanks produce errors. The uppercase characters in the commands listed as follows denote the acceptable abbreviations for the commands.

For example, to display a list of all active threads for the EDBC server, when the subsystem ID is EDBC, the command is:

```
F  EDBC, D  A
```

The following table summarizes this range of commands and the purposes they serve:

Command	Purpose	Parameters	Options
ACTivate	<p>Activates the specified protocol server for the EDBC server and makes it available to accept incoming communications.</p> <p>Example: F EDBC,ACT,PROT= TCP_KNET</p>	PROT=	CCI SNA_LUO SNA_LU62 TCP_KNET TCP_IBM TCP_SNS
INACTivate	<p>Cancels the specified thread, which can be a user ID, user context, or protocol server.</p> <p>Example: F EDBC,INACT,ID=ABC00 If a specified EDBC user ID is inactivated, the user is gracefully shutdown, with an appropriate screen prompt. All other users and tasks remain active.</p> <p>Example: F EDBC,INACT,CONTEXT= 02d08c10 If a specified user context is inactivated, the user with that unique OS/390 identifier is gracefully shut down. Other users with the same user ID but different contexts remain active.</p> <p>Example: F EDBC,INACT,PROT=SNA_LU0 If one protocol server is shut down, all threads associated with it are also shut down. If another protocol server is installed, the latter remains active, along with the entire EDBC server address space.</p>	ID= CONTEXT= PROT=	EDBC User ID Context SNA_LU0 SNA_LU62 TCP_KNET TCP_IBM TCP_SNS CCI
TRace	<p>Turns on tracing to provide dynamic information about the activity of the specified protocol server for debugging.</p> <p>Example: F EDBC,TR,PROT=SNA_LU0</p> <p>Note: Do not turn on tracing except when recommended by Technical Support.</p>	PROT=	CCI SNA_LU0 SNA_LU62 TCP_KNET TCP_IBM TCP_SNS

Command	Purpose	Parameters	Options
NOTRace	Turns off tracing for the specified protocol server. Example: F EDBC,NOTR,PROT= TCP_KNET	PROT=	CCI SNA_LU0 SNA_LU62 TCP_KNET TCP_IBM TCP_SNS
DUMP	Dumps the entire address space for troubleshooting purposes. Example: F EDBC,DUMP,ASID=14 Note: Use this command only when recommended by Technical Support.	ASID=	Numeric identifier for OS/390 address space.
HELP	Provides a display of all the gateway operator commands available for use by the OS/390 operator, with syntax and parameters. Example: F EDBC,HELP	None	None
Display Active	Provides information on all active threads in the EDBC server address space. Example: F EDBC, D A Note: See the next section for more detailed information on this command.	None	None

The Display Active Command

The following is an example of what displays on your screen if you entered the command to display the active threads for your subsystem:

```
F EDBC,D A
IEDBC: Display of Active Threads: 866
* Context Protocol Service User I/O# Inact Status *
|-----|
| 07A26010 Local Operator
| 07A26410 Local Hkeeping
| 07A26810 Local Namesvr
| 07A26C10 Local IIS1
| 07A27010 Local Monitor
| 07A27410 SNA_LU0 ProtSrvr 0 0 *ACTiv*
| 07A27810 SNA_LU62 ProtSrvr 0 0 *INACTv*
| 07A27C10 TCP_IBM ProtSrvr 0 0 *ACTiv*
| 07A28010 TCP_SNS ProtSrvr 0 0 *INACTv*
| 07A28410 CCI ProtSrvr 0 0 *INACTv*
| 07A28810 Local Hotmtr
| 07A28C10 SNA_LU0 Listener 0 1 *Listen*
| 07A29010 TCP_IBM VSAM caidem1 29 0 141.202.231.20..3658
| 07A29410 TCP_IBM VSAM caidem1 53 0 141.202.231.20..3660
| 07A29810 TCP_IBM Listener 0 0 *Listen*
| Threads (Max=31, Curr=15, User=2)
+ Timeout (Inactv=120, Force=5)
```

The following table describes the fields in this display:

Field	Description	Predefined Value
Context	<p>Unique 8-character identifier used to identify a gateway thread.</p> <p>Since more than one user can share an EDBC user ID, the context gives you a way to identify an individual user.</p> <p>Each internal thread in the gateway is assigned a unique token. Use the token to cancel a thread via the following operator command:</p> <p>F EDBC,INACT,CONTEXT=</p>	None

Field	Description	Predefined Value
Protocol	Protocol associated with the thread.	<ul style="list-style-type: none"> ■ Local - Internal Thread ■ CCI ■ SNA_LU0 - SNA LU0 protocol ■ SNA_LU62 - SNA LU6.2 protocol ■ TCP_IBM - IBM TCP/IP protocol ■ TCP_KNET- KNET TCP/IP protocol ■ TCP_SNS - Interlink SNS/TCP protocol
Service	<p>Descriptive name of each gateway thread.</p> <p>There is one <i>uuuuuu</i> entry for each remote connection.</p>	<ul style="list-style-type: none"> ■ Operator – Operator thread ■ Hkeeping – Utility thread ■ Name – Name server thread ■ EDBC – Initialization thread ■ *Listen* – Listener thread ■ uuuuuu – Name of remote user ID ■ DB2 – DB2 gateway thread ■ IMS – IMS gateway thread ■ HOTMNTR – Hot connection monitor ■ HOTCON – Gateway hot connection
User	Server class of a gateway thread. This field is valid only for gateway threads.	<ul style="list-style-type: none"> ■ VSAM – Gateway thread ■ IMS – IMS gateway thread
Net I/O	<p>Total number of sends and receives processed by the gateway from the beginning of the session.</p> <p>Shows the level of gateway activity. This field is valid only for gateway threads. The range of values is 0 to 9999999.</p>	None
Inact	Length of time, in minutes, since the last	None
	/ h f ld b d d	

Field	Description	Predefined Value
	I/O. This field can be used to determine which threads are idle. Indicates how long a *Listen* thread has been outstanding for the protocol server.	
Status	Provides current STATUS of the thread.	<ul style="list-style-type: none"> ■ IP address and port number ■ *LISTEN ■ TERM_PND* ■ NO_PSXCOM
Threads	<p>Provides summary information for the EDBC server.</p> <p>Max= Maximum number of gateway threads allowed.</p> <p>Curr= Number of active threads for this gateway. This includes both internal and gateway threads.</p> <p>User= Number of active gateway threads for this gateway.</p> <p>InactTmout= Inactivity timeout value. If this parameter is set to a value > 0, then it indicates the length of time a gateway thread is idle before it is terminated by the gateway. If it is set to 0, no timeout occurs.</p>	

Logging and Recovery

The Vantage gateway does not provide logging or recovery.

Maintaining Areas

The gateway areas contain the standard catalogs, which are crucial to the operation of the gateway. Both the standard catalogs and user table registrations are kept in the areas. For this reason, backups of the areas should be made based on the activity in the areas, such as table registrations. Periodic compresses may also be required to delete unused temporary tables allocated by Star and other internal processes.

The Vantage gateway uses linear data sets to define gateway areas. To back up and restore an area, use the standard IBM utilities, which can be invoked through the IGWFDDBA panels. In addition, a number of gateway utilities are available to assist in the maintenance of these areas. The functions provided include:

- Analyzing the space utilization of a gateway area
- Backing up a gateway area
- Restoring a gateway area
- Compressing a gateway area

Use the IGWFDDBA utilities to invoke these utilities. For more detailed information on using these functions with the IGWFDDBA panels, see the Option 2.6: Create JCL to Invoke DI Utilities Panel section in the “Using the IGWFDDBA Utilities” chapter.

The maximum size of a gateway area is 256,376,832 bytes. This corresponds to:

- 125,184 2 KB pages (DI allocation)
- 62,594 4 KB pages (VSAM linear data set allocation)
- 417 cylinders on 3380 DASD *Do not* allocate a gateway area any larger than these values.

Showing Space Allocation in an Area

The `diatedmp` utility shows space allocation information, including the 2 KB pages contained in the gateway area and a breakdown for each partition within the area. To run the `diatedmp` utility:

1. Invoke the IGWFDDBA Main Menu as described in The IGWFDDBA Main Menu section in the “Using the IGWFDDBA Utilities” chapter.
2. From the IGWFDDBA Main Menu, enter **2.6.1** in the Selection field and press Enter.

The terminal displays a panel similar to the following one:

```

tpx.cai.com
----- EDBC/Gateway Create JCL to run DIATEDMP Utility -----
                                USERID - MURTO01
                                TIME   - 14:07
                                SYSTEM - XAE1

Enter the required parameters for DIATEDMP utility:

Server => ISUREDBC      /*Logical symbol of EDBC Server */
Inarea => SABEDEM0      /*Logical symbol of User Area  */
Prefix => EDBC.USAMGW   /*EDBC files prefix           */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE
F7=UP        F8=DOWN        F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEUE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
          9/20          |CAPS          |11.11.44          |IBM-3278-2
  
```

3. For the INGAREA parameter, type the logical symbol member name that contains the name of the gateway area data set you want to analyze.
4. Press Enter to see the JCL the utility generates.
5. Review the JCL and customize it if necessary.
6. Type submit in the command line to submit the JCL and analyze the gateway area.

The report is written to ddname SYSPRINT.

Showing File, Partition, and Area Directory Information

The didirdmp utility dumps the directory associated with a specific partition. It shows the DI-files associated with a partition and the associated control blocks. In addition, you can dump a specific DI-file by specifying its name to the utility.

To run the didirdmp utility:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section of the “Using the IGWFDDBA Utilities” chapter.
2. From the IGWFDDBA Main Menu, enter **2.6.2** in the Selection field and press Enter.

The terminal displays a panel similar to the following one:

```

tpx.cai.com
----- EDBC/Gateway Create JCL to run DIDIRDMP Utility
USERID - MURTO01
TIME - 14:08
SYSTEM - XAE1

Enter the required parameters for DIDIRDMP utility:

Server  => ISUREDBC      /*Logical symbol of EDBC Server */
Ingarea => SABEDEM0     /*Logical symbol of User Area */
Prefix  => EDBC.USAMGW  /*EDBC files prefix */
Partition => IIUPUSM   /*Partition Name */
File Name =>           /*Specific DI-file name to dump */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )

F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE
F7=UP        F8=DOWN        F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEUE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
9/20 | CAPS | 11.12.51 | IBM-3279-2

```

3. For the INGAREA parameter, type the logical symbol member name that contains the name of the gateway area data set you want to analyze.
4. For the PARTITION parameter, specify the name of the partition within the gateway area you want to analyze.
5. The FILE NAME parameter is optional. Specify it when you want to dump a specific DI-file.
6. Press Enter to see the JCL the utility generates.
7. Review the JCL and customize it if necessary.
8. Type **submit** in the command line to submit the JCL and analyze the gateway area.

The report is written to ddname SYSPRINT.

DI-File Analysis Scripts

The dibackup utility produces SQL scripts with the SQL_REPORT or COMPRESS option. Several scripts are provided that allow analysis of DI-file use in a partition. The information these scripts provide includes:

- Total 2 KB pages used in a gateway area
- Total 2 KB pages used and allocated in each partition
- Total 2 KB pages used by each data manager table

- Total 2 KB pages not associated with a specific relational data manager table
- Cross-reference of the partition name, table name, and table owner to the DI-file name and DI-file extension

The scripts are located in data set EDBC.V2R3.FILES.SQL.

The following table describes the relevant members in this data set:

Script	Description
IIDICING	<p>Creates two tables in a gateway partition. The scripts generated by the dibackup utility use these tables, which are:</p> <ul style="list-style-type: none"> ■ DI_ALLOCATION Contains all DI-files that were backed up by the dibackup utility. ■ DI_DELETE_FILES Contains all DI-files that were not backed up with the COMPRESS option in dibackup. <p>When you use this script, it alters the results of the analysis because DI-pages are allocated when these tables are created.</p>
IIDIRVTA	<p>Creates two VSAM clusters as Vantage gateway VTA tables in a gateway partition. The scripts generated by the dibackup utility use these tables. The tables are:</p> <ul style="list-style-type: none"> ■ DI_ALLOCATION Contains all DI-files that were backed up by the dibackup utility. ■ DI_DELETE_FILES Contains all DI-files that were not backed up with the COMPRESS option in dibackup. <p>When you use this script, it will not alter the results of the analysis because DI-pages are not allocated when these tables are created.</p>
IIDIQDET	<p>Uses the tables created by the IIDICING or IIDIRVTA scripts, and is loaded by the output from dibackup SQL_REPORT or dibackup COMPRESS. This script produces a detailed report showing how the DI-pages are allocated in the gateway area.</p> <p>This script shows only the cross-reference between the relational tables and the DI-files in the partition that is being accessed.</p>

Script	Description
IIDIQSUM	<p>Uses the tables created by the IIDICING or IIDIRVTA scripts, and is loaded by the output from dibackup SQL_REPORT or dibackup COMPRESS. This script produces a summary report showing how the DI-pages are allocated in the gateway area.</p> <p>This script does <i>not</i> generate a cross-reference.</p>

Procedure

You can analyze the use of space in an area by performing the following steps:

1. Run dibackup using either the SQL_REPORT or COMPRESS option. (For more information, see the “Using the IGWFDDBA Utilities” chapter.)
2. Save the SYSPRINT output from the dibackup job into a sequential file.
3. Copy this file to a member in the EDBC.V2R3.FILES.SQL data set. You can call this member IIDILxxx, for example.
4. Connect to the gateway partition using the Terminal Monitor.
 - a. Use the IIDICING or IIDIRVTA script to create the required tables.
 - b. Use the IIDILxxx script to load the information produced in the dibackup utility into the DI analysis tables.
 - c. Use the IIDIQSUM script to obtain a summary of page usage for the gateway area.
 - d. Use the IIDIQDET script to obtain a detailed analysis of page usage for files associated with tables in that partition.

Backing Up an Area

The dibackup utility allows you to specify and back up a gateway area. To use this utility:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section of the “Using the IGWFDDBA Utilities” chapter.
2. From the IGWFDDBA Main Menu, enter **2.6.3** in the Selection field and press Enter.

The terminal displays the EDBC/Gateway Create JCL to Run DIBACKUP Utility panel.

3. For the INGAREA parameter, type in the logical symbol member name that contains the name of the area data set to be analyzed.

For example, type:

```
SABEDEM0
```

4. For the BACKUP parameter, type the name you are assigning to the backup data set.

For example, with the default data set prefix, type:

```
EDBC.V2R3.INGAREA.DEMO.BACKUP
```

5. Leave the OPTIONS parameter blank, to specify a standard backup operation.

The completed panel appears as shown in the following figure:

6. Press Enter to see the JCL the utility generates.
7. Review the JCL and customize it if necessary.

If the gateway area or the sort work data set you are backing up does not have the default size, modify the JCL to enter the correct values.

8. Type **submit** to submit the JCL and back up the gateway area.
All return codes should be 0 or 4. The last step should be 0.
9. When the area is backed up, press PF3 repeatedly to return to the Vantage Gateway Main Menu.

Restoring an Area

Note: If the area being restored is in use by the Vantage gateway, the gateway must be brought down before you run the restore job.

Follow these steps to restore a gateway area with the direstor utility:

1. Invoke the IGWFDBA Main Menu as described in the IGWFDBA Main Menu section of the “Using the IGWFDBA Utilities” chapter.

- From the IGWFDBA Main Menu, enter **2.6.4** in the Selection field and press Enter.

The terminal displays the Gateway Facility to Create JCL to Run DIRESTOR Utility panel.

- For the BACKUP parameter, type in the name you assigned to the backup data set.

For example, with the default data set prefix, type:

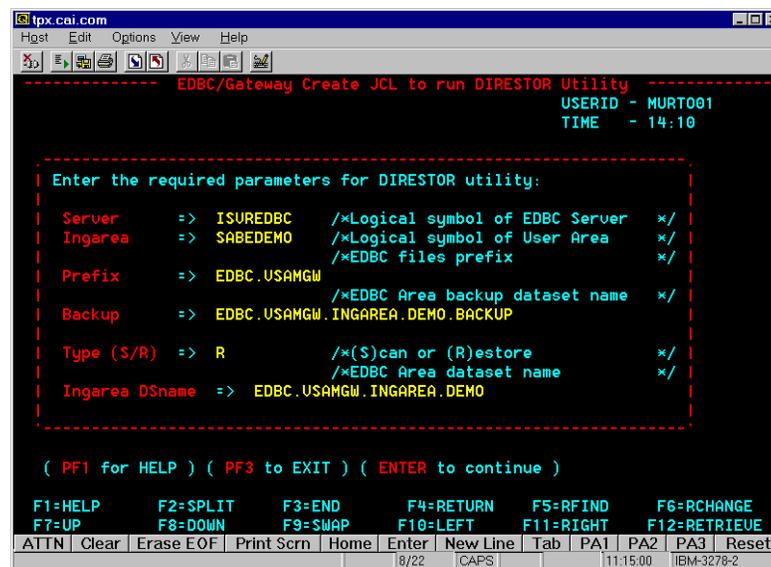
```
EDBC.V2R3.INGAREA.DEMO.BACKUP
```

- For the TYPE parameter, type **R** to specify a restore operation.
- For the INGAREA DSNAME parameter, type in the data set name of the area you want to restore.

For example, with the default data set prefix, type:

```
EDBC.V2R3.INGAREA.DEMO
```

The completed panel appears as follows:



- Press Enter to see the JCL the utility generates.
- Review the JCL, and customize it if necessary.

If the gateway area or the sort work data set that you are restoring does not have the default size, modify the JCL to enter the correct values.

- Submit the JCL to restore the gateway area.

All return codes should be 0 or 4. The last step should be 0.

- When the area is restored, press PF3 repeatedly to return to the Vantage gateway Main Menu.

Increasing the Size of an Area

To increase the size of an area:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section of the “Using the IGWFDDBA Utilities” chapter.
2. From the IGWFDDBA Main Menu, enter **2.6.3** in the Selection field and press Enter.

The terminal displays the Gateway Facility to Create JCL to Run DIBACKUP Utility panel.

3. Specify the appropriate parameter values and run the backup operation.

For more information on the parameters, see the Option 2.6.3: The Gateway Facility to Create JCL to Run Dibackup Utility Panel section in this chapter.

4. Return to the EDBC/Gateway Create JCL to Invoke DI Utilities panel.
5. Type 4 in the Selection field and press Enter to select the direstor utility.
6. Specify appropriate parameters for the restore.

For more information, see the Option 2.6.4: The Gateway Facility to Create JCL to Run Direstor Utility Panel section.

7. Press Enter to see the JCL the utility generates.

In the ALLOCARA step, increase the space allocation to the desired number of 2 KB blocks. When this job is submitted, it deletes, allocates, and copies the backed up gateway area into the larger newly allocated gateway area.

Compressing an Area

Periodic compresses of the gateway area may be required due to the specific internal temporary tables that are not released automatically. The dicomprs utility is a combination of the dibackup and direstor utilities with specific parameter settings to do a compress.

You can use the analysis utilities such as diatedmp and didirdmp to determine when a compress is required. Note that an area cannot be compressed at the same time it is being accessed by a gateway.

To run the dicomprs utility:

1. Invoke the IGWFDDBA Main Menu as described in the IGWFDDBA Main Menu section of the “Using the IGWFDDBA Utilities” chapter.
2. From the IGWFDDBA Main Menu, enter 2.6.5 in the Selection field and press Enter.

The terminal displays a panel similar to the following one:

```

tpx.cai.com
----- EDBC/Gateway Create JCL to run DICOPYAR Utility
USERID - MURTO01
TIME   - 14:13

Enter the required parameters for DICOPYAR utility:

Server   => ISUREDBC   /*Logical symbol of EDBC Server */
Prefix   => EDBC.USAMGW /*EDBC files prefix */

Source Area => SABEDEM0 /*Logical symbol of Source Area */
Target Area => UCAIDEM1 /*Logical symbol of Target Area */
Target User => CAIDEM1 /*Userid for Target User */
Target DSname => EDBC.USAMGW.INGAREA.CAIDEM1 /*EDBC Area dataset name */

( PF1 for HELP ) ( PF3 to EXIT ) ( ENTER to continue )
F1=HELP  F2=SPLIT  F3=END    F4=RETURN  F5=RFIND  F6=RCHANGE
F7=UP    F8=DOWN   F9=SWAP   F10=LEFT  F11=RIGHT F12=RETRIEVE
ATTN | Clear | Erase EOF | Print Scrn | Home | Enter | New Line | Tab | PA1 | PA2 | PA3 | Reset
----- 8/22 | CAPS | 11:17:35 | IBM-3278-2
  
```

3. For the SOURCE AREA parameter, specify the logical symbols member name that contains the name of the source gateway area data set. For example, type:
SABEDEM0
 4. For the TARGET AREA parameter, specify the logical symbols member name that contains the name of the target user data set. For example, type:
UCAIDEM1
 5. For the TARGET USER parameter, specify the user ID for the target user. For example, type:
CAIDEM1
 6. For the TARGET DSNAME parameter, specify the data set name of the target gateway area. For example, type:
EDBC.V2R3.INGAREA.CAIDEM1
 7. Press Enter to see the JCL the utility generates.
 8. Review the JCL and customize it if necessary.
 9. Type **submit** in the command line to submit the JCL.
- The report is written to ddname SYSPRINT.

Using the Gateway Utilities

The gateway provides the following utilities for managing gateway areas and partitions:

- createpr
- dbconst
- destroypr
- finddb
- igwzfp
- verifydb
- vwconst

This section provides reference information on these utilities. For information on the command syntax, see the "Introduction" chapter in this guide.

You access all these utilities except igwzfp by invoking the DBMS Utility Menu option of the IGWFDDB utilities. You access igwzfp through the ISPF EDIT option.

createpr

The createpr utility creates a new partition with the specified name, in the area of the EDBC user ID whose logicals are currently set.

Syntax

```
createpr -dbname [-u] [-m] [-vantageipaddr:port]
[-maxcalls] -f
```

Description

The parameters for this command are given in the following table:

Parameter	Status	Description
-dbname	Required	Specifies the name of the partition (or database) to be created.

Parameter	Status	Description
-u	Optional	Specifies the intent of the partition's access to the Vantage objects as update. If -u is not specified, the partition is read-only. Note that updates are not supported for the initial release of Vantage support.
-m	Optional	Takes the specified parameters and bypasses the create partition logic, merely updating the external gateway catalog: For Vantage-iigw01psb (If -v option is specified)
-vvantageipaddr:port	Required	Specifies the TCP/IP address and port number of the Vantage subsystem associated with this partition.
-cmaxcalls	Optional	Indicates the maximum number of gateway calls that can be made per transaction. The default is unlimited.
-f	Optional	Specifies to invoke the upgradefe utility to upgrade the frontend catalogs in the partition. This eliminates the need to run upgradefe as a separate step. You generally specify this option when you create new partitions, but you do not need it when you modify (-m option) a partition.

Example

The following are examples of the createpr utility:

```
createpr -dvants21-f -v255.1.1.1:999
```

This example creates a new partition VANTS21 for Vantage read access. The TCP/IP address and port number for the associated Vantage subsystem is 255.1.1.1:999.

dbconst

The dbconst utility performs consistency checking on the specified gateway area.

Syntax

```
dbconst ing_area
```

Description

Note: The gateway area cannot be in use. If it is in use at the time, dbconst waits for the area to free up before a consistency check is performed.

The parameter for this command is listed in the following table:

Required Value	Description
ing_area	Logical symbol for the gateway area. Refer to the “Gateway Logical Symbols and IIPARM Clist” appendix for a more detailed description of how this logical symbol is constructed.

destroyp

The destroyp utility destroys a user partition.

Syntax

```
destroyp dbname [-s] [-uusername] [-p]
```

Description

Destroying a gateway partition removes the partition from the standard system catalogs for that gateway area, and destroys all the table templates, EDBC application objects, and other data registration information for the Vantage objects that are accessed through that partition. The utility then performs a consistency cleanup of the gateway area linear data set. The process has no effect on the underlying Vantage data.

To use this utility you must be the user who created the partition or be an EDBC superuser. Destroyp cannot be used to destroy the gateway database (iidbdb).

The parameters for this command are listed in the following table:

Parameter	Status	Description
<i>dbname</i>	Required	Specify the name of the partition to be destroyed.
-s	Optional	Allows an EDBC superuser to destroy any partition.
-uusername	Optional	Allows an EDBC superuser to destroy a partition owned by the specified <i>username</i> .
-p	Optional	Asks whether you really want to destroy the specified partition.

finddb

The finddb utility rebuilds the iidbdb master database for a given installation.

Syntax

```
finddb [-a] [-r] [-p] [-v]
```

Description

The parameters for this command are listed in the following table:

Parameter	Description
-a	Runs finddb in analyze mode. This is the default mode. In this mode, gateway areas are searched for a default partition. After all areas are searched, a report lists: <ul style="list-style-type: none">■ The partitions found in these gateway areas that were not entered in the iidatabase table of the system catalog for iidbdb.■ The databases entered in the iidatabase table in the system catalogs that were not found in the areas searched.
-r	Runs finddb in replace mode. In this mode, gateway areas are searched for default partitions. After all areas have been searched, rows are appended to the iidatabase relation in the iidbdb for every partition that was found in the search.

Parameter	Description
-p	When running in replace mode, marks the rows in the iidatabase relation as private partitions.
-v	Reports more diagnostic information to the user.

igwfzap

The igwfzap utility performs special system upgrades to the IGWF subsystem and reports on system activity. To access this utility, use the ISPF EDIT option. For the data set name, type:

EDBC.V2R3.SAMPLE.CNTL(IGWFZAP)

where *EDBC.V2R3* is the gateway data set prefix.

Description

The igwfzap utility should be used only by the gateway DBA after consultation with Technical Support. The utility is included here for purposes of technical reference.

This utility is executed using the following JCL:

```
//IGWFZAP JOB (EDBCDBA,CNS), 'VANTAGE.GW.IGWFZAP',
// MSGCLASS=H,MSGLEVEL=(1,1),
// NOTIFY=INGDBA,REGION=4M,
// CLASS=A,USER=EDBCDBA
/**
/**
/** DEFINE THE CATALOGED PROCEDURE THAT WILL REINSTALL
/** THE IGWF SUBSYSTEM MODULES. THE LOAD LIBRARY
/** REFERENCED IN THIS PROCEDURE **MUST** BE MVS
/** APF-AUTHORIZED
/**
//IGWFZAP      PROC REQCODE=,PREFIX='EDBC.V2R3'
//IGWFZAP      EXEC PGM=IGWFZAP,PARM='&REQCODE'
//STEPLIB      DD DSN=&PREFIX..BACK.LOAD,DISP=SHR
//IGWFZAPD     DD DSN=&PREFIX..BACK.LOAD,DISP=SHR
//SYSPRINT     DD SYSOUT=*
//SYSUDUMP     DD SYSOUT=*
//SYSOUTDD     SYSOUT=*
//             PEND
/**
/** INSPECT THE IGWF SUBSYSTEM TO GET CURRENT STATUS
/**
//INSPECT EXEC IGWFZAP,REQCODE=INSPECT
/**
/** INSTALL THE FOLLOWING IGWF SUBSYSTEM MODULES
/** DYNAMICALLY
/**
//GWFFC00      EXEC IGWFZAP,REQCODE=GWFFC00
//GWFFC01      EXEC IGWFZAP,REQCODE=GWFFC01
//GWFFC00      EXEC IGWFZAP,REQCODE=GWFFC00
```

```

//GWFFC01      EXEC IGWFZAP,REQCODE=GWFFC01
//GWFFC02      EXEC IGWFZAP,REQCODE=GWFFC02
//GWFFC03      EXEC IGWFZAP,REQCODE=GWFFC03
//GWFFC08      EXEC IGWFZAP,REQCODE=GWFFC08
//GWFFC10      EXEC IGWFZAP,REQCODE=GWFFC10
//GWFFCI0      EXEC IGWFZAP,REQCODE=GWFFCI0
//GWFFCI0      EXEC IGWFZAP,REQCODE=GWFFCI1
//IGWFAG       EXEC IGWFZAP,REQCODE=IGWFAG
//GWFSCHK      EXEC IGWFZAP,REQCODE=GWFSCHK
//DD@PC13      EXEC IGWFZAP,REQCODE=DD@PC13
//DD@PC22      EXEC IGWFZAP,REQCODE=DD@PC22
//DD@PC31      EXEC IGWFZAP,REQCODE=DD@PC31
//DD@PC41      EXEC IGWFZAP,REQCODE=DD@PC41
//DD@PC51      EXEC IGWFZAP,REQCODE=DD@PC51
//DD@PC61      EXEC IGWFZAP,REQCODE=DD@PC61
//*
//* RESET DCCB CHAIN
//*
//DCCB         EXEC IGWFZAP,REQCODE=DCCB
//*
//* INSPECT THE IGWF SUBSYSTEM TO GET CURRENT STATUS
//*
//INSPECT      EXEC IGWFZAP,REQCODE=INSPECT
    
```

The following table lists the optional parameters in this syntax:

Parameter	Description	Special Options
inspect	Inspects the gateway environment and creates a report using 370/Assembler formatted DSECTS. This report shows all active drivers on the system and address locations as well as maintenance levels of various gateway components.	370/C: Generates the report using 370/C data structures instead of 370/Assembler DSECTS.
dccb	Refreshes the gateway XCSA memory. This reinitializes the IGWF DCCB chain only if there are no active drivers running on the system. If reinitialization is required and this option fails, make sure all drivers have been stopped before restarting.	DCCB also provides an option to clean up individual DCCB links on the chain. This option is: DCCB -Axxxxxxx -SCAN where: <ul style="list-style-type: none"> ■ Axxxxxxx =the DCCB link address within the DCCB chain. You can obtain this address by first executing the 'INSPECT' option. ■ SCAN=An optional parameter used to scan the DCCB link before actually performing the cleanup.

Parameter	Description	Special Options
force	<p><i>Caution!</i> This option should be used only if all others fail. If this option fails, an OS/390 IPL must be done to refresh the DCCB chain.</p> <p>Like the DCCB options, this option refreshes the gateway environment. It resets the SSCT anchor by zeroing out the first DCCB chain link.</p>	None

verifydb

The verifydb utility operates on partitions to assure their consistency.

Syntax

```
verifydb -mmode -sscope -ooperation [-uuser]
        [-ddebug] [-rrel] [-nnolog]
```

Description

Verifydb:

- Forces an inconsistent partition to appear consistent to the dbms
- Performs checks and, optionally, associated repairs, on DBMS catalogs

The following table lists the parameters of the verifydb utility:

Parameter	Description	Special Options
-mmode	Specifies the mode in which verifydb runs. Use this mandatory flag with one of the following qualifiers:	<ul style="list-style-type: none"> ■ report – No actions are performed: only diagnosis and reporting of what should be done in any of the run modes. ■ run – Perform the actions and notify the user accordingly. ■ runsilent runs – Perform actions without notifying the user. ■ runinteractive runi – Determine the action to take, then prompt the user for permission before taking that action.

Parameter	Description	Special Options
<code>-sscope</code>	Specifies the scope of the <code>verifydb</code> action. Use this mandatory flag with one of the following qualifiers:	<ul style="list-style-type: none"> ■ <code>dbname</code> – Operates on the databases specified on the input line. Note: All databases specified in this list must be owned by the same user. ■ Up to 10 database names can be specified. The syntax is: <code>-sdbname "DB1 DB2 DB3 ... DB10"</code> ■ <code>dba</code> – Operates on all databases for which the user is the database administrator. An EDBC superuser may use this qualifier with the <code>-user</code> flag. ■ <code>installation</code> – Operates on all of an installation's local databases.
<code>-ooperation</code>	Specifies the <code>verifydb</code> operation to be performed. Use this mandatory flag with the following qualifiers:	<ul style="list-style-type: none"> ■ <code>dbms_catalogs dbms</code> – DBMS system catalogs are checked, and if any of the run qualifiers are selected, these are corrected. ■ <code>force_consistent force</code> – Makes inconsistent database look consistent to the DBMS. This option only works with a single database at a time. ■ <code>drop_tbl</code> – Drops a table from the system catalogs without checking or dropping the disk files associated with the table. (Must specify the name as well.)
<code>-uuser</code>	Used by a gateway DBA or an EDBC superuser to enter the database as a specific user.	None
<code>-ddebug</code>	Cause informative messages to be output to the screen for debugging purposes.	None
<code>-rrel</code>	Run only the portion of the system catalog checks that clears the optimizer	None

Parameter	Description	Special Options
	statistics information.	
<code>-nolog</code>	Skip opening and writing to the verifydb log file (not recommended).	None

Example

An example of the verifydb utility is:

```
verifydb -mreport -sdbname "VANTS21" -odbms -ddebug
```

vwconst

The vwconst utility performs consistency checking for VIEW corruption on the specified partition (database) and deletes inconsistent views.

Syntax

```
vwconst dbname
```

Description

The parameter for this command is shown in the following table:

Parameter	Status	Description
dbname	Required	Specifies the name of the partition to be checked and corrected.

Defining an Additional EDBC Server

There may be times when you need to create an additional EDBC server to Vantage on the same OS/390 host for load balancing, to allow access to another Vantage Multi-User Facility, or for resource control. The following procedure describes how to use the IIVP to define the second EDBC server.

Using IIVP to Create a Second EDBC Server

Follow these steps to create a second EDBC server using the IIVP:

1. Define a second gateway subsystem name to OS/390.
Each server must have a unique OS/390 subsystem defined.
2. Add this new subsystem name to the IEFSSNxx member in SYS1.PARMLIB.
You must IPL OS/390 at the end of this procedure to activate this new subsystem.
3. Define the network changes needed to access this second server.
 - SNA LU0 protocol server
Define the VTAM APPL for this server. You can duplicate the existing VTAM APPL and change the name.
 - TCP/IP protocol server
Select the Port ID for this server. This Port ID must be different from the original Port ID assigned to the first server.
 - CCI protocol server
Select the CCI Product ID for this server. This value must be different from the original Product ID assigned to the first server.
4. Edit the stage1 input file IGWFSTGT in the data set with the default name EDBC.V2R3.FILES.ASM.
5. Make the following changes to each SYSGEN statement:

IGWFJOB:	It is recommended that you change the JOBNAME= prefix so that the members that are placed into the STAGE2.CNTL data set have unique names and do not replace previous members. Use the new installation ID as part of the prefix.
IGWFVSAM:	No changes required. PREFIX= Set this parameter to the data set prefix assigned to the VSAM data sets.

IGWFINET:	<p>Change the following parameters as indicated:</p> <p>SUBSYS=</p> <p>Set this parameter to the value of the new gateway subsystem name.</p> <p>ID=</p> <p>Set this parameter to the value of the new II_INSTALLATION value.</p> <p>MAXUSER=</p> <p>Set this parameter to the maximum number of users to be allowed on the system.</p>
IGWFPSVR:	<p>Add a new IGWFPSVR statement corresponding to the network protocol parameters that will be needed to access the new server. Use the following guidelines for each parameter:</p> <p>TYPE=</p> <p>Set this to the type of protocol server that the new gateway will use.</p> <p>ACB=</p> <p>If the LU0 protocol server is being used, set this value to the new VTAM APPL defined for this server.</p> <p>PLU=</p> <p>If the KNET protocol server is being used, set this value to that of the KNET primary Logical Unit value.</p> <p>PORT=</p> <p>If the TCP/IP protocol is being used, set this value to that of the new port ID associated with this server.</p> <p>INSTALL=</p> <p>Set this value to that of the IGWFINET ID= value.</p> <p>APPLID=</p> <p>If the Interlink SNS/TCP protocol server is being used, set this value if access to the SNS/TCP subsystem is restricted by the application id.</p> <p>PASSWORD=</p> <p>If the Interlink SNS/TCP protocol server is being used, set this value if a password is required to interface to the SNS/TCP subsystem.</p>

SYSID=

If the Interlink SNS/TCP protocol server is being used, set this value if the subsystem name of the SNS/TCP subsystem is different than the default of ACSS.

PRODID=

If the CCI protocol server is being used, set this value to that of the new CCI Product ID associated with this server.

IGWFBLD: Make the following change to this SYSGEN statement:

PRODUCTS= (EDBC)

This specifies that the IIVP stage1 is to create a jobstream that defines a new server, but does not install any new gateway system catalogs in VSAM.

6. Save the IGWFSTGT member in the data set with the default name EDBC.V2R3.FILES.ASM.
7. Edit the member IGWFSTGT in the data set with the default name EDBC.V2R3.SAMPLE.CNTL and submit it for execution.

This job creates the IIVP stage2 jobstream in a data set with the default name EDBC.V2R3.STAGE2.CNTL.

8. Review the jobstream in this data set and submit it for execution.

The jobstream consists of the following jobs:

IGWVFPA0: Creates logical symbols to customize the gateway

IGWVFPI0: Customizes Net

9. Release each job sequentially.

These jobs define a second EDBC server to Vantage. See the Final Installation Procedures section in the "Installing the Gateway" chapter for the post-installation steps that you must complete.

Connecting from One EDBC Server to Another

When you want to connect from one EDBC server to a second EDBC server, use the procedure described in this section.

When you have created a second EDBC server, the second server (EDC2) must be defined to act as a client of the first server (EDBC). You must complete the following procedure in order to allow a remote client connected to the second EDBC server to connect to the first EDBC server.

1. Verify that both servers (EDBC and EDC2) have been started.
2. From ISPF option 6, issue the following command to set up the logical symbols to access the second server (EDC2).

```
%iiparm isvr(edc2) prefix(edbc.v2r3)
```

This TSO Clist allocates the IIPARM DD statement that contains the logical symbols that allow the TSO address space to connect to the EDC2 server.

3. Type the following to invoke the netu utility:

netu

The netu utility is used to define the information required by a client to connect to a remote server, in this instance, INT2.

4. At the NETU prompt, type **n** to select Modify Node Entry and press Enter.
5. At the following prompt:
Enter operation (add, del, show):
type **add** and press Enter.
6. Type the following responses to the following prompts that appear:

NETU Prompt	Response
Enter Private or Global (P):	p
Enter the remote v_node name:	EDBCUSER01
Enter the network software type:	TCP_IBM
Enter the remote node address:	<i>Internet Address</i>
Enter the remote Net server listen address:	<i>port ID</i>

The following message displays:

```
Private: 1 row(s) added to the Server Registry
```

7. Type the following to return to the netu menu:
/*
8. At the NETU prompt, type **a** to select Modify Remote Authorization Entry.
9. Type the following responses to each of the following prompts that appear:

NETU Prompt	Response
Enter Private or Global (P):	p
Enter the remote v_node name:	EDBCUSER01
Enter the remote User Name:	EDCUSR1

NETU Prompt	Response
Enter the remote Password:	xxxxxxx
Repeat the remote Password:	xxxxxxx

The following message appears:

Private: 1 row(s) added to the Server Registry

9. Return to the netu menu by typing `/*`:
10. Type `e` to exit from netu.

Once you complete the steps above, a Net v_node entry with a corresponding remote user ID entry is defined to access the EDBC server, EDBC.

Ensure that the remote user name is a valid Vantage authorization ID, if required, and the associated password is valid.

Verifying the Connection

To verify that you can connect to the first EDBC server (EDBC) from the second server (EDC2), complete the following steps:

1. Verify that both servers are started.
2. Type the following to access EDC2 from ISPF option 6 from a TSO terminal:
`%iiparm isvr(edc2) pre(edbc.v2r3)`
3. Issue the following command to invoke the Net/Terminal Monitor:

sql edbcuser01::vsam/vsam

The Net/Terminal Monitor prompt should appear, indicating that you have successfully connected to the first gateway.

The standard catalog, iidbconstants, can be used to verify that the correct user ID is being used to access the gateway. Issue the following command from the Terminal Monitor:

select * from iidbconstants\g

Using the sample values, the following response should appear:

USER_NAM	DBA_NAME	SYSTEM_O
EDCUSR1	EDCUSR1	\$EDBC

(1 row)

The value in the USER_NAM column should match what was specified in the remote user name definition.

The Gateway Query Optimizer

The gateway uses a query optimizer to develop sophisticated query execution strategies. The query optimizer makes use of basic information such as row size, number of rows, primary key fields and indexes defined.

Without knowing exactly what data you have stored in your table, the query optimizer uses default assumptions about what your data looks like. Consider the following examples:

```
select * from emp
  where gender = 'M';

select * from emp
  where empno = 13;
```

In query 1, a good guess is that a column entitled gender probably has two possible data values. Another guess is that a company would consist of approximately 50% males and 50% females. However, the gateway does not know how restrictive a where clause like this is. Without any other information, the gateway would incorrectly assume that this clause was very restrictive.

In query 2, the gateway again can only guess how restrictive this where clause is. Without knowing that employee numbers are usually unique, the gateway guesses that few rows qualify; in this case, this is the correct assumption.

Why does it make any performance difference how restrictive the gateway thinks your query is? For a single-table keyed retrieval where you are specifying the key, there is probably no difference at all. The gateway uses the key to retrieve your data. However, in a multi-table query with several restrictions, knowing what your data looks like can help the gateway guess the best way to execute your query. The following example shows why:

```
select e.name, e.dept, b.address
  from emp e, dept d, bldg b
  where e.dept = d.dname
  and d.bldg = b.bldg
  and b.state = 'CA'
  and e.salary = 50000;
```

There are many ways of executing this query. If appropriate keys exist, the gateway probably chooses to execute the query in one of these two ways:

- Retrieve all the employees with a salary of 50000. Join the employees with a salary of 50000 to the department table, and then join the employees with their valid departments to the valid buildings. The tables are processed in the following order:

```
emp -> dept -> bldg
```

- Retrieve all the buildings with a state of CA. Join the valid buildings with the department table, and join the qualifying departments to the valid employees. The tables are processed in the following order:

```
bldg -> dept -> emp
```

The difference between these two options is that #1 first joins employees to departments, then departments to buildings, and #2 first joins buildings to departments and then departments to employees.

Which method is preferable? Only if you knew exactly how many employees made \$50,000, how many buildings were in California, and how many departments were in each building would you be able to pick the best strategy.

The best query execution strategy, that is, the fastest strategy, can only be determined if the gateway has some idea what your data looks like: How many rows qualify from the restriction, and how many rows join from table to table.

Query execution plans (QEPs) illustrate how the gateway executes a query. Knowing how to read and analyze QEPs can allow you to detect, and often avoid, hidden performance problems.

The Query Execution Plan (QEP)

When the query optimizer evaluates a query it generates a QEP showing how the query will be executed. Once the QEP has been generated, the gateway can use it one or more times to execute the corresponding database query. Since there are often many different ways to optimize any given query, choosing the best QEP may have a significant impact on performance.

You can print an informal listing of the QEP selected, which can be used to gain insight into how queries are handled by the gateway query optimizer.

The word query refers to any of the data manipulation language (DML) statements that require access to the database. These include the SQL statements select, insert, update, delete, and create table as.

Examining QEPs help you understand what is involved in executing complex queries in single-user situations.

Listing a QEP

In general it is a good idea to test run a query in the terminal monitor. You can obtain a printed copy of the QEP, using the set optimizeonly statement to generate a QEP without actually executing the query.

Use one of the following statements to list QEPs:

```
set qep (Terminal Monitor statement)
exec sql set qep; (embedded SQL)
```

To view query execution plans without executing the queries, use set optimizeonly in conjunction with set qep.

Less frequently you might want a QEP from a query submitted through a program. In this case, define an environment variable/logical that the gateway will translate:

UNIX

C shell:

```
setenv ING_SET "set qep"
```

Bourne shell:

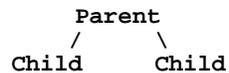
```
ING_SET = "set qep" export ING_SET ?
```

VMS

```
define ING_SET "set qep" ?
```

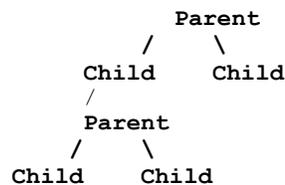
Reading a QEP

A QEP is printed as a binary tree, with each node having one or two children:



The tree is read in postorder sequence (that is, child, child, parent recursively). Of the different types of nodes (described in the next section) only Join and Cart-Prod nodes have two children; other nodes have only a left child.

Different levels in the tree are also separated by lines containing "\" and "/" symbols:



Each level has one or more nodes, with each node being made up of a block of 3 to 5 lines of text (explained below). As noted, the nodes are connected by "/" which is a left child link and "\" which is a right child link. The "/" and "\" links point to a node one level up in the tree.

When reading a QEP the first thing to do is to find the "orig" nodes of the tree, which are those nodes with no children. Orig nodes contain details of the tables (and secondary indexes) that are being used in the query.

Queries that have been "modified" by the gateway to include views, integrities and/or permits are more involved. For very involved queries, the tree may be very wide and could wrap so that similar levels in the tree appear on different levels in the printout. You may find it easier to read such QEPs if you cut and paste the diagrams into the correct levels.

Sample Tables for QEP Examples

In the examples in this section, the following two tables are used:

1. Table arel(col1,col2,col3):

```

Name:                are1
Owner:               supp60
Created:             26-oct-1989 08:50:00
Location:            ii_database
Type:                user table
Version:             ING6.0
Row width:           413
Number of rows:      156
Storage structure:   hash
Duplicate Rows:      allowed
Number of pages:     70
Overflow data pages: 6
Column Information:
Column
Name      Type      Length  Nulls  Defaults  Key
Seq
col1      integer   4       yes    no         1
col2      integer   4       yes    no
col3      varchar  400     yes    no
Secondary indexes:  aindex (col2)      structure: isam
    
```

2. Table brel(col1,col2):

```

Name:                brel
Owner:               supp60
Created:             26-oct-1989 08:53:00
Location:            ii_database
Type:                user table
Version:             ING6.0
Row width:           10
Number of rows:      156
Storage structure:   isam
Duplicate Rows:      allowed
Number of pages:     3
Overflow data pages: 1
Column Information:
Column
Name      Type      Length  Nulls  Defaults  Key
Seq
col1      integer   4       yes    n         1
col2      integer   4       yes    no
Secondary indexes:  none
    
```

Types of Nodes in a QEP

There are three general types of nodes on a QEP:

- Orig (or leaf) node – describes a particular table

- Proj-rest node – describes the qualification, or projection restriction, on one or more subsidiary orig nodes
- Join nodes – describes a join. One of the following strategies is used:
 - Cartesian product
 - Full sort merge
 - Partial sort merge
 - Key and tid lookup joins
 - Subquery joins

Also, many of these nodes may be shown as a sorted node. This type of node is also described below.

The entries that may appear on a QEP are shown in the following node parameters table:

Parameter	Description
tablename	The table on which the query is being run, usually a table name. It is a secondary index if secondary indexes are selected by the query optimizer for execution of the query.
storage structure	The storage structure in use: Btree(key NU) Hashed(key NU) Heap Isam(key NU) where key is the key. If the primary key will not be used, the notation NU (Not Used) appears.
colname	The name of the column on which sorting is done. For colnames there may be a list of column names separated by blanks.
Pages n Tups m	n is the total number of pages involved at the node (1 page = 2 KB bytes). m is total number of tuples (rows).
Dx Cy	Cumulative amounts of cost that are anticipated at each stage in the execution of the query:
	Dx Disk I/O cost. x approximates the number of 2K pages to be referenced.
	Cy CPU usage. y units can be used to compare amounts of CPU resources required.

Orig Node

The orig, or leaf, node usually describes a table being accessed from the query. This node is displayed in the following format on the QEP listings:

tablename storage structure (colname) Pages n Tups m

Orig nodes are the most common type of QEP node. The orig node lists the storage structure of the tables being used by the query and the key, if any (unless the type is heap, which does not have a parenthesized entry).

Proj-rest Node

The project-restrict (proj-rest) node defines how a subsidiary orig node is to key into the table and what qualification to use on the table. This node is displayed in the following format on the QEP listings:

Proj-rest Heap Pages n Tups m Dx Cy or Proj-rest Sort on (colnames) Pages n Tups m Dx Cy

Proj-rest nodes are used to remove data irrelevant to the query from a table so that the minimum amount of data is carried around during the execution of the query. Only those columns referenced in the target list and where clause for a table are projected, and only those rows that satisfy constraints in the where clause are restricted for use.

Proj-rest nodes mostly consume disk operations that read the data from the tables. The amount of disk I/O depends on what storage structures were used as well as the number of pages accessed.

For proj-rest and other non-orig nodes, the storage structure generally is heap unless the data is being sorted.

All non-orig nodes also show the cumulative count line:

Dx Cy

Since these values are cumulative up the tree, the top node carries the total resources required to execute the query. The effort involved in executing a specific node is therefore the D and C values for that node minus those of the child node (or both child nodes in the case of a join node).

Sort Nodes

Sort nodes cause the data to be sorted as it is returned. Any node other than an orig node can appear with a sort indication. A query with a sort clause on it has a sort node as the topmost node in the QEP. This node is displayed in the following format on the QEP listings:

Sort on(colname) Pages n Tups m Dx Cy

If the sort node is combined with a proj-rest node, the sort columns are displayed along with proj-rest.

Sort nodes make use of a sort buffer and so consume primarily cpu resources, except requiring disk I/O when the data to be sorted cannot be accommodated in the sort buffer. The gateway uses the Quicksort algorithm, which is very fast on nearly sorted data. The gateway query optimizer tends to sort whenever the data fits in the sort buffer (even if it believes only one row will be sorted).

Examples

Some QEP examples that illustrate these non-join nodes are shown below.

This is an example of a simple primary key lookup. The query execution plan is shown below for the following SQL query:

```
select col1,col2 from are1
  where col1 = 1234
  order by col2;
*****
QUERY PLAN 3,1, no timeout, of main query
      Sort Keep dups
        Pages 1 Tups 1
        D1 C0
      /
      Proj-rest
        Sorted(col1)
        Pages 1 Tups 1
        D1 C0
      /
      are1
        Hashed(col1)
        Pages 70 Tups 156
*****
```

In the preceding example the Hashed(col1) implies the hashed primary index was used in a direct lookup (that is, only 1 disk operation unit). The project restrict selected the rows matching the where constraint and removed superfluous columns. The final sort node reflects the sort clause on the query.

The following is an example of a select on a non-keyed field. The query execution plan is shown below for the following SQL query:

```
select col1,col2 from arel
  where col3 = 'x'
  order by col1;
*****
QUERY PLAN 3,1, no timeout, of main query
      Sort Keep dups
        Pages 1 Tups 1
        D9 C0
        /
        Proj-rest
        Heap
        Pages 1 Tups 1
        D9 C0
        /
        arel
        Hashed(NU)
        Pages 70 Tups 156
*****
```

In this example the Hashed(NU) implies that the table had to be scanned (i.e., visiting all 70 pages). Without optimizedb statistics, the gateway query optimizer uses a "best guess" approach (1% for equalities and 10% for non-equalities).

Also note that the gateway query optimizer takes into account disk readahead or group reads when performing scans of tables; although 70 pages of data have to be read to scan arel, the estimated disk I/O value is only 9 reads (D9) due to this effect. The gateway query optimizer assumes a typical readahead of 8 pages when performing scans, so here 70/8 reads generates an estimate of 9 disk operations.

Join Nodes

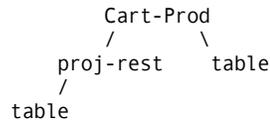
There is an inner and an outer tree beneath every join node. These function similarly to an inner and outer program loop. By convention, the left-hand tree is the outer tree and the right-hand tree is the inner tree.

There are various types of join nodes, described individually below, but the joining method is the same: For each row from the outer tree, there is a join to all of the rows that can possibly qualify from the inner tree. Then the next row from the outer tree is processed, and so on.

Any join node can have outer join information if an outer join is present.

Cartesian Products

The cartesian product, or "cart-prod," strictly follows the unoptimized join model, with each row in the outer node compared to all rows from the inner node.



This does not mean that all rows are actually read, only that all rows that satisfy the conditions of the query are compared. This node is displayed in the following format on the QEP listings:

Cart-Prod (colname) [LEFT JOIN] [FULL JOIN] [RIGHT JOIN] heap Pages n Tups m Dx Cy

The cart-prod is most frequently observed in disjoint queries (that is, when use of correlation variables and table names are mixed in the query). However, the following cases may also generate cart-prods without adversely affecting performance:

- Queries involving ORs that can usually be decomposed into a sequence of smaller queries
- No join specification (a disjoint table or no where clause so that there is no relationship between the two tables)
- Small tables or amounts of data
- Non_equijoins:

where r1.col1 > r2.col2

Cart-prods are sometimes associated with substantial estimates for disk I/O usage and affect performance adversely.

Here is an example of a simple disjoint query:

```

select arel.col1 from arel,arel a
  where a.col1 = 99;
*****
QUERY PLAN 7,1, no timeout, of main query
      /      \
    Cart-Prod
     /        \
  Proj-rest   Proj-rest
  Sorted(NU)  Heap
  Pages 1 Tups 1  Pages 1 Tups 156
  D1 C0        D8 C1
   /          /
  arel       arel
  Hashed(col1) Hashed(NU)
  Pages 70 Tups 156  Pages 70 Tups 156
*****
  
```

Full Sort Merge

The full sort merge is an optimized join that attempts to join the inner and outer trees doing fewer comparisons than the cart-prod requires. This is done by assuming that both trees are in sorted order. If they are not, they are sorted. This allows each outer value to be joined to a contiguous subset of inner values rather than going through the entire inner table.



This node is displayed in the following format on the QEP listings:

FSM join (colname) [LEFT JOIN] [FULL JOIN] [RIGHT JOIN] Heap or Sorted(a)
 Pages n Tups m Dx Cy

The full sort merge is most common when a bulk join takes place with no row restrictions on either table involved in the join. The SQL statement would have the form:

select * from r1,r2 where r1.joincol=r2.joincol;

Here is an example:

```

select a.col2,b.col2 from arel a, brel b
  where a.col1 = b.col1;
*****
QUERY PLAN 5,1, no timeout, of main query
      FSM Join(col1)
      Heap
      Pages 1 Tups 156
      D9 C40
     /          \
  Proj-rest      Proj-rest
  Sorted(eno)    Sorted(eno)
  Pages 1 Tups 156  Pages 1 Tups 156
  D8 C1           D1 C1
  /              /
arel            brel
Hashed(col1)    Isam(col1)
Pages 70 Tups 156  Pages 3 Tups 156
*****
    
```

As an example of the subselect flattening enhancement features of the gateway query optimizer, consider the following subselect:

```

select r.a from r where r.c =
  (select avg(s.a) from s where s.b = r.b and r.d > s.d)
    
```

Instead of two scans of table r, the gateway query optimizer has eliminated a scan of table r by evaluating the aggregate at an interior QEP node.

The query plan appears similar to the previous example:

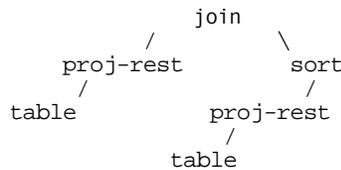
```

*****
QUERY PLAN 7,1, no timeout, of main query
      FSM Join(b)
      avg(s.a)
      Heap
      Pages 2 Tups 17
      D133 C171
     /      \
  Proj-rest  Proj-rest
  Sorted(b)  Sorted(b)
  Pages 1 Tups 359  Pages 40 Tups 359
  D65 C112      D32 C3
  /            /
 r            s
Btree (b,c)   Hashed(b)
Pages 44 Tups 359  Pages 44 Tups 359
*****

```

Partial Sort Merge

The partial sort merge is a cross between a full sort merge and a cart-prod. The inner tree must be in sorted order. The outer tree may be sorted or partially sorted. Note that the outer tree in PSM scenarios will always be derived from an isam table. Comparisons proceed as for the full sort merge until an outer value is found to be out of order. At that point the inner loop is restarted.



This node is displayed in the following format on the QEP listings:

```

PSM join (colname) [LEFT JOIN] [FULL JOIN] [RIGHT JOIN] Heap Pages n
  Tups m Dx Cy

```

Here is an example:

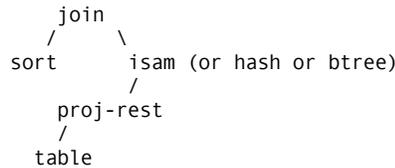
```

select a.col2,b.col2 from arel a, brel b
  where a.col1 = b.col2;
*****
QUERY PLAN 6,1, no timeout, of main query
      PSM Join(col1)
      Heap
      Pages 1 Tups 156
      D9 C26
     /      \
  Proj-rest  Proj-rest
  Heap      Sort on(col1)
  Pages 1 Tups 156  Pages 1Tups 156
  D1 C1      D8 C1
  /          /
 brel      arel
  Isam(col2) Hashed(col1)
  Pages 3 Tups 156  Pages 70 Tups 156
*****

```

Key and Tid Lookup Joins

In key and tid lookup joins the outer and inner data set is not static. For each outer row, the join selects values and forms a key to search in the inner join. A key lookup join uses keyed access, and a tid lookup join uses the tid value.



This node is displayed in the following format on the QEP listings:

K | T join (colname) [LEFT JOIN] Heap Pages n Tups m Dx Cy

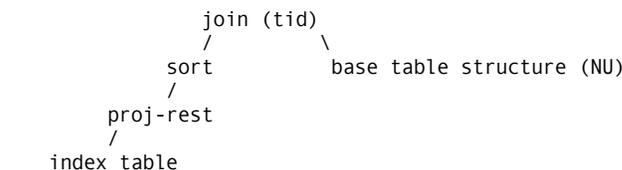
This case is seen most frequently where the proj-rest returns few rows, so it is faster to do a keyed lookup on an isam (hash or btree) table than any sort merge operations.

Here is an example:

```

select b.col1,b.col2,a.col2
  from arel a, brel b
  where a.col3 = 'x' and a.col1 = b.col1;
*****
          K Join(col1)
          Heap
          Pages 1 Tups 1
          D3 C0
         /      \
    Proj-rest      brel
    Heap          Isam(col1)
    Pages 1 Tups 1    Pages 3 Tups 156
    D1 C0
     /
    arel
    Hashed(col1)
    Pages 70 Tups 156
*****
    
```

In the next example, access to the base table is through the secondary index and proj-rest collects tids for sorting. The tids are then used for a direct tid lookup in the base table. Hence the storage structure of the base table is NU.



Here is an example:

```

select a.col1,a.col2 from arel a
  where a.col2 = 99
  order by a.col2;
    
```

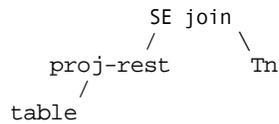
```

*****
                Sort(col2)
                Pages 1 Tups 1
                D4 C1
                /
                T Join(tidp)
                Heap
                Pages 1 Tups 1
                D4 C0
                /
                Proj-rest          \ arel
                Sort on(tidp)      Hashed(NU)
                Pages 1 Tups 1      Pages 70 Tups 156
                D2 C0
                /
                aindex
                Isam(col2)
                Pages 2 Tups 156
*****

```

Subquery Joins

The subquery join is specific to SQL because SQL allows subselects as part of a query. These nodes join data to the subselect specified in the query.



where Tn identifies the previous QEP.

This node is displayed in the following format on the QEP listings:

SE join (colname) Heap Pages n Tups m Dx Cy

Here is an example:

```

select * from arel a
  where arel.col2 = (
    select col2 from brel b
    where a.col1 = b.col1)
  and col1 = 5;
*****
QUERY PLAN 3,1, no timeout, of T1
      Proj-rest
      Heap
      Pages 1 Tups 1
      D1 C0
      /
      brel
      Hashed(col1)
      Pages 3 Tups 156
QUERY PLAN 4,2, no timeout, of main query
      SE Join
      Heap
      Pages 1 Tups 1
      D2 C0
      /
      Proj-res      \
      Heap          T1
      Pages 1 Tups 1  Heap
      D1 C0          Pages 1 Tups 1
      /
      arel
      Hashed(col1)
      Pages 70 Tups 156
*****

```

Multiple Query Plans

The gateway query optimizer may generate multiple QEPs if the query includes any of the following objects:

- SQL subqueries (in, exists, all, any , and so on.)
- SQL union clause
- SQL group by clause
- Views that need to be materialized

As an example of multiple QEPs, consider the processing of a view on a union. The following statement creates the view:

```

create view view1 as
  select distinct col1 from arel
union
  select distinct col2 from arel;

```

There are two selects, designated #1 and #2 in their respective QEPs below.

Now consider the gateway query optimizer action in evaluating the following query:

```
select * from view1;
```

This generates three QEPs:

- #1 - the first select in the union
- #2 - the second select in the union
- Main query

```
*****
QUERY PLAN of union subquery #1
      Sort Unique
      Pages 1 Tups 156
      D1 C10
    /
    Proj-rest
    Heap
    Pages 1 Tups 156
    D1 C0
  /
  aindex
  Isam(col2)
  Pages 2 Tups 156
QUERY PLAN of union subquery #2
      Sort Unique
      Pages 1 Tups 156
      D4 C10
    /
    Proj-rest
    Heap
    Pages 1 Tups 156
    D4 C0
  /
  are1
  Hashed(NU)
  Pages 70 Tups 156
QUERY PLAN of main query
      Sort Unique
      Pages 1 Tups 156
      D19 C20
    /
    Proj-rest
    Heap
    Pages 1 Tups 156
    D12 C11
  /
  T0
  Heap
  Pages 1 Tups 156
*****
```

More Complex QEPS

The previous series of QEPs on the different classes of joins involved only two tables. More complex QEPs involving joins with three or more tables can be read as a sequence of two-table joins that have already been described, with the gateway query optimizer deciding what is the optimal join sequence. The key to understanding these complex QEPs is recognizing the join sequences and the types of joins being implemented.

Evaluating QEPS: A Summary

Here is a list of the main points to check for when evaluating a QEP:

- Cart-Prods can be caused by errors due to disjoint queries or queries that involve certain types of "or" operations. Also, joins involving calculations, data type conversions and non-equijoins involve cart-prods. Alternative ways of posing the query is often advised under these circumstances.
- The NU on the storage structure part in the orig node description is not a good sign if you believe the storage structure should have been used.
- Verify that the appropriate secondary indexes are being used.
- If there is little data in a table (e.g., less than 5 pages) the gateway query optimizer may consider a scan of the table rather than use any primary or secondary indexes, because little is to be gained from using them
- Check that row estimates are accurate on the QEP nodes.

Optimizer Timeout

The gateway query optimizer does not search for further execution plans if it believes that the best plan it has found so far would take less time to execute than the time it has taken evaluating query plans. In this case the gateway query optimizer times out, stopping its operation and returning with the best query execution plan it currently has.

You can tell if the gateway query optimizer timed out by checking the header of the QEP. For normal query plans, which find the optimal QEP without timing out, the QEP looks similar to:

```
QUERY PLAN n,m, no timeout, of main query
```

However queries that timed out before searching all query plans have a header similar to:

```
QUERY PLAN n,m, timed out, of main query
```

You can turn the timeout feature off in order to enable the gateway query optimizer to evaluate all plans. You can use any of the following statements:

```
set joinop notimeout           (Terminal monitor statement)
exec sql set joinop notimeout; (Embedded SQL)
```

UNIX

```
C Shell:      setenv ING_SET "set joinop notimeout"
              (operating system)
```

```
Bourne Shell: ING_SET = "set joinop notimeout";
              export ING_SET (operating system) ?
```

VMS

```
define ING_SET "set joinop notimeout"
              (operating system) ?
```

You can reset the default action of optimizer timeout by using the timeout parameter, as in the following example:

```
set joinop timeout
```

Since "time" above is elapsed cpu time, QEPs that time out can change, depending on machine load.

Note: The fact that the gateway query optimizer has timed out on a particular query does not guarantee that a better query plan could be found; it is an indication that not all query plans have been checked, and so potentially a better query plan could be found.

Debugging the Gateway

This section outlines the debugging aids that are available to the gateway DBA. You can also use these tools to determine the location and cause of any abnormalities in gateway operation.

To invoke the trace facility requires the EDBC Terminal Monitor program, running under TSO or as a batch application. The EDBC Terminal Monitor allows the setting of trace points at the CSECT or module level. The trace facilities may generate a large volume of output. It is advisable to run tracing in a batch environment or to limit the number of queries issued.

All trace output is sent to the data set specified by the IIERLOG DD statement.

Tracing the Gateway Routing Exits

To set trace points within the Gateway Routing Exits, use the following commands:

```

For module GW_DEFINE          enter: set trace point dm01324\g
For module GW_OPEN            enter: set trace point dm01325\g
For module GW_CLOSE          enter: set trace point dm01326\g
For module GW_POSITION       enter: set trace point dm01327\g
For module GW_PUT             enter: set trace point dm01328\g
For module GW_GET             enter: set trace point dm01329\g
For module GW_REPLACE        enter: set trace point dm01330\g
For module GW_DELETE         enter: set trace point dm01331\g
For module GW_ABORT          enter: set trace point dm01337\g
    
```

To turn off these trace points, use the following commands:

```

For module GW_DEFINE          enter: set notrace point dm01324\g
For module GW_OPEN            enter: set notrace point dm01325\g
For module GW_CLOSE          enter: set notrace point dm01326\g
For module GW_POSITION       enter: set notrace point dm01327\g
For module GW_PUT             enter: set notrace point dm01328\g
For module GW_GET             enter: set notrace point dm01329\g
For module GW_REPLACE        enter: set notrace point dm01330\g
For module GW_DELETE         enter: set notrace point dm01331\g
For module GW_ABORT          enter: set notrace point dm01337\g
    
```

Tracing the Gateway Components

To set trace points within the gateway components, use the following commands:

```

For module DEFINE            enter: set trace point dm01303\g
For module OPEN              enter: set trace point dm01304\g
For module CLOSE             enter: set trace point dm01305\g
For module POSITION           enter: set trace point dm01306\g
For module PUT               enter: set trace point dm01307\g
For module GET               enter: set trace point dm01308\g
For module REPLACE           enter: set trace point dm01309\g
For module DELETE            enter: set trace point dm01310\g
    
```

To turn off these trace points, use the following commands:

```

For module DEFINE            enter: set notrace point dm01303\g
For module OPEN              enter: set notrace point dm01304\g
For module CLOSE             enter: set notrace point dm01305\g
For module POSITION           enter: set notrace point dm01306\g
For module PUT               enter: set notrace point dm01307\g
For module GET               enter: set notrace point dm01308\g
For module REPLACE           enter: set notrace point dm01309\g
For module DELETE            enter: set notrace point dm01310\g
    
```

Diagnosing IGWF Initialization Failures After an IPL

Abends numbered 047, 071, and 0C4 after an IPL are usually caused by a missing PCHOLD and/or SSVT pointers in the IGWF Subsystem Control Table (SSCT).

The only way this can happen is after some serious problems during subsystem initialization at IPL time. The problem can be confirmed by analyzing output from the IGWFZAP REQCODE=INSPECT utility (&prefix.sample.cntl(IGWFZAP): some of the addresses off the IGWF SSCT will point to low core (PSWs, and so on).

The following steps should help in diagnosing the reasons behind IGWF initialization failures at IPL time:

1. Start by looking at the JES log from the beginning (the IPL) and look for IG or IGW. Do you see any error messages about unauthorized libraries, modules not found, and so on?
2. If the answer is yes, resolve the errors and attempt another IPL.
3. If the answer is no, look at SYS1.PARMLIB(IEASYSxx) (xx is the value the operator specified as the SYS parameter at IPL time). In that IEASYSxx member, you will find an SSN parameter with one or more values. Each one of these is the last two characters of an IEFSSNxx member in SYS1.PARMLIB. Go through each IEFSSNxx member looking for an IGWF entry.

The entry should be IGWF,IGWFSSP. This entry indicates to OS/390 that it has to call IGWFSSP for IGWF subsystem initialization immediately after obtaining the storage for the IGWF SSCT. For IGWFSSP to get control, it has to reside in a LINKLIST library (refer to the LNK parameter in IEASYSxx: It contains the suffixes of all the relevant LNKSTxx members that will be used to construct the LINKLIST). Look at these LNKSTxx members. Do you see the library that contains all the members that were copied by STEP010 in the B0 job (see the "Installing the Gateway" chapter)?

4. There is one last thing you need to check: Issue a TSO LISTC ENT('library.name') for that LINKLIST library to verify that it is cataloged in the MASTER catalog (a library used for subsystem initialization *must* be cataloged in the MASTER catalog because subsystem initialization takes place *before* catalog address space initialization, thereby making it impossible to resolve catalog aliases).

Resubmitting the B0 job will resolve IGWF problems until the next IPL.

Gateway Logical Symbols and IIPARM Clist

Gateway behavior is controlled by logical symbols that are specified when the gateway is invoked. The IIPARM DD statement determines which logical symbols are used. In the batch environment (batch job or started task), IIPARM can point to a sequential data set or a member of a PDS.

In the foreground (TSO), the IIPARM clist supplied in the EDBC.V2R3.FILES.CLIST data set is used to specify the member of a PDS containing the logical symbols to be used for the session. The EDBC.V2R3.FILES.IIPARM data set contains default logical symbol members that are created from user stage1 macro specifications.

Logical Symbol Library Organization

The gateway Logical Symbol library has the default name EDBC.V2R3.FILES.IIPARM. This library contains members that are used for different purposes. These members are:

- ISVRxxxx member

Logical symbols to initialize or access a EDBC server. This member is created from user stage1 macro specifications.

- SABExxxx member

The VANTAGE gateway can be invoked independently of the EDBC server. In other words, the VANTAGE gateway can be run as a batch job or under TSO, even though the EDBC server is not started. This mode of operation is called Stand Alone Back End (SABE) and is determined by the logical ING_MODE. Specifying ING_MODE=NOSERVER; will enable the VANTAGE gateway to be invoked independently of the EDBC server. Specifying ING_MODE=SERVER; indicates that the batch job or TSO user will interface to the VANTAGE gateway through the EDBC server and thus requires that the EDBC server be started. In addition to the ING_MODE logical, several other logicals are required to run in Stand Alone Back End mode. These logicals are generated and stored in library EDBC.V2R3.FILES.IIPARM as member SABEDEM0 by the stage2 installation process.

- Uxxxxxxx member

These members contain the same type of logical symbols as those in the SABExxxx member, with which they can be interchanged.

The advantage of this set of members is that you can customize up to seven characters of the member name, as opposed to only four characters of the SABExxxx member set.

Logical Name Format

The gateway uses logical names to configure many operational parameters. These logicals have the following syntax:

```
LOGICAL_NAME = 'value';
```

You can customize logicals by editing the text that appears to the right of the equals (=) sign. Each record in a logical file may be:

- Blank
- An assignment, using the syntax:

```
LOGICAL_NAME = 'value';
```

where the value is expressed as a string or a number, depending on the logical name. Enclose string values in single or double quotes. Each logical name assignment must end with a semicolon.

- A comment, delineated by /* and */
- An assignment followed by a comment

Detailed Descriptions of Logical Symbols

This section describes the logical symbols you can configure. A logical symbol can be specified more than once. If this happens, the system uses the last occurrence of that logical symbol. Logical symbols are grouped into several members. This section describes the symbols that are contained in each of the following members:

- ISVREDBC – Server logical symbols
- DBNMEDBC – Remote user name to database area association
- SABExxxx – Stand-alone back end logical symbols

ISVREDBC Logical Symbol Members

These logical symbols are used to initialize the EDBC server and to provide access to that server from a remote client, a batch, or a TSO address space. The IIPARM Clist also uses these logical symbols in conjunction with symbols in the DBNMEDBC and SABExxxx members.

II_VANTAGE_MODULE This specifies the name of the Vantage EWS Interface load module. This should be set to: VANTIO

```
II_VANTAGE_MODULE = VANTIO;
```

II_BUFFERS This specifies the number of SORT buffers that will be used. The non-relational gateway products use this parameter. An example is:

```
II_BUFFERS = 20;
```

II_DATABASE This specifies the name of the logical symbol that contains the database locations. This value is fixed and must not be modified. In the non-relational gateway products, this value is used under two circumstances:

- Access from TSO or batch

The value of this logical symbol specifies the logical symbol that contains the database area name.

- Access from Net

The value of this logical symbol is prefixed to the user ID of the remote user to create the logical symbol that contains the database area name. An example is:

```
II_DATABASE = ING_AREA;
```

II_DATE_CENTURY_BOUNDARY

In the gateway, the year defaults to the current year. In formats that include delimiters (such as forward slashes or dashes), you can specify the last two digits in the year; the first two digits default to the current century (1900). For example, if you enter 03/21/93 using the format *mm/dd/yyyy*, the gateway assumes that you are referring to March 21, 1993. This behavior forces the user to specify all 4 digits of the year when dealing with dates in the next century.

The **II_DATE_CENTURY_BOUNDARY** logical symbol allows you to change this default. It contains an integer that has a value between 0 and 100 inclusive. If this logical symbol is set, and if the setting is in the valid range, then the century will be determined by the following calculation:

```
if (current_year < boundary)
  if (input_year < boundary)
    output_year = input_year + current_century
  else
    output_year = input_year + previous_century
else
  if (input_year < boundary)
    output_year = input_year + next_century
  else
    output_year = input_year + current_century
```

thus:

```
II-DATE-CENTURY-BOUNDARY= '93';
```

Then an input date of 03/31/93 will be treated as March 21, 1993. However, an input date of 03/21/03 will be treated as March 21, 2003.

II_DBMS_LOG

This specifies the location of the DBMS error log. The value can be either a ddname or a dsname.

- ddname

If a ddname is given as the value, it is in the form *DD:ddname* where *ddname* is the value of a data definition (DD) statement that was included in the EDBC server JCL.

- dsname

If a dsname is given as a value, it is in the form *dataset.name* where *dataset.name* is the fully qualified name of an existing sequential data set. This data set will be dynamically allocated and used to log database management system errors. An example is:

```
II_DBMS_LOG = 'DD:IIERLOG';
```

II_DBTMPL

This specifies the name of the non-relational gateway database template directory. This symbol must be entered exactly as shown:

```
II_DBTMPLT = 'ING_AREA:DB.iicore';
```

II_FILES

This specifies the OS/390 data set prefix of the various gateway installation data sets. If this symbol is not defined, then the gateway will concatenate the value of the SYS_INGRES logical symbol with the character string .FILES to create the data set prefix. An example is:

```
II_FILES = 'EDBC.V2R3.FILES';
```

II_FORCE_TMOUTINT

This specifies the number of minutes of inactivity, following an attempted user disconnect, an operator inactivate, or an inactivate user timeout, that the system will wait before forcing the user connection down if it still exists. This is similar to the inactive user timeout facility but is a second-stage more severe form of a "kill."

The default is 5 minutes. An example is:

```
II_FORCE_TMOUTINT = 5;
```

II_FORMFILE

This specifies a front-end parameter that defines the location of cached forms files. An example is:

```
II_FORMFILE = 'EDBC.V2R3.FILES.RTIFORMS.FNX';
```

II_GCC_ID	<p>This specifies the OS/390 subsystem name that is used by the gateway address space. This value must match the value of the II_GCN_ID symbol. An example is:</p> <pre>II_GCC_ID = EDBC;</pre>
II_GCCI1_LOG	<p>This specifies the location of the gateway communications server error log. The value given can be either a ddname or a dsname.</p> <ul style="list-style-type: none">■ ddname If a ddname is given as a value, it will be of the form DD:ddname where ddname is the value of a data definition (DD) statement that was included in the Net JCL.■ dsname If a dsname is given as a value, it will be of the form <i>dataset.name</i> where <i>dataset.name</i> is the fully qualified name of an existing sequential data set. This data set will be dynamically allocated and used to log communication server errors. If the II_INSTALLATION symbol has a value other than II, then this logical symbol will have the characters II replaced by the new specification. An example is: <pre>II_GCCI1_LOG = 'DD: IIERLOG' ;</pre>
II_GCN_ID	<p>This specifies the OS/390 subsystem name that is used by the gateway name server. This value must match the value of the II_GCC_ID symbol. An example is:</p> <pre>II_GCN_ID = EDBC;</pre>
II_GCNI1_LOG	<p>This specifies the location of the gateway name server error log. The value given can be either a ddname or a dsname.</p> <ul style="list-style-type: none">■ ddname If a ddname is given as a value, it will be of the form DD: ddname where <i>ddname</i> is the value of a data definition statement (DD) that was included in the Net JCL.■ dsname If a dsname is given as a value, it will be of the form <i>dataset.name</i> where <i>dataset.name</i> is the fully qualified name of an existing sequential data set. This data set will be dynamically allocated and used to log communication server errors. If the II_INSTALLATION symbol has a value other than II, then this logical symbol will have the characters II replaced by the new specification. An example is: <pre>II_GCNI1_LOG = 'DD: IIERLOG' ;</pre>

II_GCNI1_LCL_VNODE This specifies the virtual node name of the gateway Comm server. It is used by the name server to determine whether a remote connection or local connection is desired. If it is not present, the name server will always attempt a remote connection. If it is present, the name server compares the requested virtual node to this value. If there is a match then the gateway is started locally in the TSO or batch address space. If the **II_INSTALLATION** symbol specifies a value other than **I1**, then the symbol's name must be changed accordingly. An example is:

```
II_GCNI1_LCL_VNODE = EDBC;
```

II_GCNI1_SVR_TYPE This specifies the default server class that will be used during a connection process if a server class is not given in the connect statement. The valid values are:

- **DCOM**—specifies the default server class, the CA-Datacom/DB gateway.
- **DB2**—specifies the default server class, the DB2 gateway.
- **IDMS**—specifies the default server class, the CA-IDMS gateway.
- **IMS**—specifies the default server class, the IMS gateway.
- **VSAM**—specifies the default server class, the VSAM gateway.
- **VANT**—specifies the default server class, the Vantage gateway.

If the **II_INSTALLATION** symbol specifies a value other than **I1**, then the symbol name must be changed accordingly. An example is:

```
II_GCNI1_SVR_TYPE = VANTAGE;
```

II_GENERIC_ERROR The gateway maps VSAM-specific errors to a series of generic errors. This mapping scheme makes application-error-handling portable across different gateway products and platforms.

The valid values are **YES** and **NO**. If the logical symbol is defined as **YES**, when the gateway detects an error in the SQL statement execution, it returns the following:

- **EDBC OpenSQL Generic Error Code**
- **VSAM Error Code**

If the logical is defined as **NO**, the Data Manager returns the standard error messages. An example is:

```
II_GENERIC_ERROR = YES;
```

II_HELPDIR This specifies the data set that contains the Terminal Monitor help files. The help files are members of the data set with the default name **EDBC.V2R3.FILES.HLP**, which was part of the installation tape. An example is:

```
II_HELPDIR = "PDS: 'EDBC.V2R3.FILES.HLP'";
```

II_INACTV_TMOUTINT	<p>This specifies the number of minutes of inactivity after which user threads are timed out. User threads that may be “hung” as a result of users breaking out of front-end applications (for example, pressing PA1 in TSO) are automatically released after the specified timeout. Security is enhanced because unattended sessions are automatically logged out. Gateway threads and resources are freed up.</p> <p>If the parameter is not specified, or if it is set to zero, the timeout facility is not activated – that is, no automatic timeouts will occur. If you specify the parameter but do not assign it a value, it defaults to 0. An example is:</p> <pre>II_INACTV_TMOUTINT = 0;</pre>
II_INSTALLATION	<p>This specifies the 2-character code that is the gateway installation ID. The default value for this code is I1. It must be unique for each EDBC server address space.</p> <p>This 2-character code appears in the following logical symbols:</p> <pre>II_GCCii_LOG II_GCNii_LCL_VNODE II_GCCii_SVR_TYPE</pre> <p>where ii is the value specified in the II_INSTALLATION symbol.</p> <p>This installation code can also appear in several logical symbols that are not normally used:</p> <pre>II_GCCii_LOGLVL II_GCCii_ERRLVL</pre> <p>This installation code, along with the value of the II_FILES logical symbol, is used to create the OS/390 data set names that comprise the name server database.</p> <p>These files are:</p> <pre>EDBC.V2R3.FILES.NAME.IICOMSVR.ii EDBC.V2R3.FILES.NAME.IIDB2.ii EDBC.V2R3.FILES.NAME.IIIMS.ii EDBC.V2R3.FILES.NAME.IIVANTAGE.ii EDBC.V2R3.FILES.NAME.IIINGRES.ii EDBC.V2R3.FILES.NAME.IILOGIN.ii EDBC.V2R3.FILES.NAME.IINODE.ii EDBC.V2R3.FILES.NAME.IIVANTAGE.ii</pre> <p>where ii is the value of the II_INSTALLATION logical symbol.</p> <p>An example of the default entry is:</p> <pre>II_INSTALLATION = I1;</pre>
II_NET_LMOD	<p>This logical symbol specifies the name of the IIPSERV (protocol server) load module to use for Net initialization. The default value is IIPSERV.</p> <pre>II_NET_LMOD = 'IIPSERV';</pre>

II_NO_ENQ_SUBSYS	<p>This specifies that the gateway be invoked within the Net address space. It must be specified as follows:</p> <pre>II_NO_ENQ_SUBSYS = YES;</pre>
II_PSF_POOL	<p>This specifies the number of 4 KB pages that should be allocated to the Parser Facility (PSF). This parameter should be set to 30 to support Microsoft Access and Visual Basic clients. If this value is set too low, clients receive message "E_RD0001 Not enough memory for RDF temporary work area". The default value is 25 4 KB pages. An example of the default entry is:</p> <pre>II_PSF_POOL = 25;</pre>
II_PROTOCOL_CCI	<p>The valid values are YES and NO. If the logical number is defined as YES, it specifies that the installation is using the CCI protocol for OS/390 to OS/390 communications. An example is:</p> <pre>II_PROTOCOL_CCI = YES;</pre>
II_PROTOCOL_RESTART_COUNT	<p>The number of times the EDBC server should attempt to restart a protocol driver after a protocol driver failure. The valid values are 0-10. Default is 0.</p>
II_PROTOCOL_SNA_LU0	<p>The valid values are YES and NO. If the logical symbol is defined as YES, it specifies that the installation is using the SNA LU0 protocol for communications between the IBM and the remote EDBC system. An example is:</p> <pre>II_PROTOCOL_SNA_LU0 = YES;</pre>
II_PROTOCOL_SNA_LU62	<p>The valid values are YES and NO. If the logical symbol is defined as YES, it specifies that the installation is using the SNA LU62 protocol for communications between the IBM and the remote EDBC system. An example is:</p> <pre>II_PROTOCOL_SNA_LU62 = YES;</pre>
II_PROTOCOL_TCP_IBM	<p>The valid values are YES and NO. If the logical symbol is defined as YES, it specifies that the installation is using the IBM TCP/IP protocol for communications between the IBM and the remote EDBC system. An example is:</p> <pre>II_PROTOCOL_TCP_IBM = YES;</pre>
II_PROTOCOL_TCP_IBM_BUFSIZE	<p>The IBM TCP/IP packet size in kilobytes. Valid values are 4-32. Default is 8. An example is:</p> <pre>II_PROTOCOL_TCP_IBM_BUFSIZE = 16;</pre>
II_PROTOCOL_TCP_KNET	<p>The valid values are YES and NO. If the logical symbol is defined as YES, it specifies that the installation is using the KNET TCP/IP protocol for communications between the IBM and the remote EDBC system. An example is:</p> <pre>II_PROTOCOL_TCP_KNET = YES;</pre>

II_PROTOCOL_TCP_SNS	<p>The valid values are YES and NO. YES specifies that the installation is using the Interlink SNS/TCP protocol for communications between OS/390 and the remote EDBC system. An example is:</p> <pre>II_PROTOCOL_TCP_SNS = YES;</pre>
II_PROTOCOL_TCP_SNS_BUFSIZE	<p>The SNS TCP/IP packet size in kilobytes. Valid values are 4-32. Default is 8. An example is:</p> <pre>II_PROTOCOL_TCP_SNS_BUFSIZE = 16;</pre>
II_QEP_SIZE	<p>This specifies the maximum memory used by the Query Execution Facility (QEF) to build data segment headers. The Query Execution Facility manages and executes query plans of non-relational gateway products, and contains the information required by QEF to execute the query. The default is 3072 bytes. An example is:</p> <pre>II_QEP_SIZE = 3072;</pre>
II_RECALL	<p>This specifies whether the EDBC server should retrieve any archived data sets, through an exit, prior to completing its initialization. The sample JCL to compile and link IIRECALL exit is in EDBC.V2R3.SAMPLE.CNTL (ASMRCALL).</p> <p>The valid values are YES and NO. If the logical symbol is defined as YES, it invokes the IIRECALL exit.</p> <p>Note: The IIRECALL exit must be compiled and linked into EDBC.V2R3.BACK.LOAD using sample JCL.</p> <p>The default value is NO. An example is:</p> <pre>II_RECALL = NO;</pre>
II_SECURITY	<p>This specifies the type of OS/390 host security used by the EDBC server. The values that can be specified are CA-ACF2, RACF, CA-TSS, or NONE. An example is:</p> <pre>II_SECURITY = RACF;</pre> <p>If no value or an invalid value is specified, then the default is NONE.</p>
II_SMFID	<p>This specifies the value of the SMF record number that will be generated during the writing of user accounting records. This value should always be set to 0. If this value is 0, it disables the writing of SMF records. If a value is given that falls between 128 and 255 inclusive, then an SMF accounting record is written using this record ID.</p> <p>Note: This function has not been enabled for the current release of the gateway products. This information is provided for planning purposes. An example is:</p> <pre>II_SMFID = 0;</pre>

II_TIMEZONE	<p>This specifies the difference in hours between Greenwich Mean Time (GMT) and the local time zone where the EDBC server is running. If this logical is not specified, it defaults to 0. An example is:</p> <pre>II_TIMEZONE = 7;</pre>
II_UTEXE_DEF	<p>This specifies the location of front-end initialization parameters. This file must be allocated and available during execution of gateway user interfaces on OS/390. It is distributed as an empty file. It is recommended that the symbol be entered as follows:</p> <pre>II_UTEXE_DEF = "'EDBC.V2R3.FILES.UTEXE.DEF'";</pre>
ING_MODE	<p>This specifies the execution mode for gateway programs. There are two values that are normally used. These are:</p> <ul style="list-style-type: none">■ SERVER<p>This specifies that the gateway initializes in server mode. It is used in the following circumstances:</p><ul style="list-style-type: none">- Net initialization.<p>Used when the Net address space is initializing.</p>- Local front-end access from TSO or Batch.<p>Used when a front end wishes to locally access the EDBC server address space. Needed by utilities such as iinamu and netu to access the gateway name server.</p>■ NOSERVER<p>This specifies that the gateway initializes in no-server mode. It is used in the following circumstances:</p><ul style="list-style-type: none">- CREATPR processing.<p>This mode is set in the SABExxxx member of the logical symbol library for formatting and initializing database areas and partitions.</p>- TSO or Batch access to the non-relational gateway products.<p>This mode is set if single-user access to the non-relational gateway products is desired.</p> <p>The following shows the value that is specified in the ISVRxxxx member for this symbol.</p> <pre>ING_MODE = SERVER;</pre>
LOCK_LISTS	<p>This specifies an internal non-relational gateway parameter. It must be entered as follows:</p> <pre>LOCK_LISTS = 2047;</pre>

LOCK_MAX	<p>This specifies an internal non-relational gateway parameter. It must be entered as follows:</p> <pre>LOCK_MAX = 32768;</pre>
LOCK_RESTAB	<p>This specifies an internal non-relational gateway parameter.</p> <p>It must be entered as follows:</p> <pre>LOCK_RESTAB = 2047;</pre>
LOCK_TABSIZE	<p>This specifies an internal non-relational gateway parameter. It must be entered as follows:</p> <pre>LOCK_TABSIZE = 2047;</pre>
LOG_CPINTVL	<p>This specifies an internal non-relational gateway parameter. It must be entered as follows:</p> <pre>LOG_CPINTVL = 4;</pre>
LOG_SBLKS	<p>This specifies an internal non-relational gateway parameter. It must be entered as follows:</p> <pre>LOG_SBLKS = 48;</pre>
MAX_LOCKS	<p>This specifies an internal non-relational gateway parameter. It must be entered as follows:</p> <pre>MAX_LOCKS = 350;</pre>
SRV_MAXSERVERS	<p>This specifies the number of gateway utility sub-tasks that will be created to perform functions required by the EDBC server. This number should be set equal to at least the number of 370 central processors on OS/390. The default value is:</p> <pre>SRV_MAXSERVERS = 4;</pre>
SRV_MAXTHREADS	<p>This specifies the maximum number of concurrent threads that the gateway address space can support. The system reserves the first 16 threads for its own use. Each local connection requires a single thread. Each remote connection requires a single thread. Thus, specify the maximum number of expected connections plus 16 to arrive at this value. The default value of 64 will allow 48 concurrent user connections to be established. The default value is:</p> <pre>SRV_MAXTHREADS = 64;</pre>
SRV_STKSIZE	<p>This specifies an internal gateway parameter. Specify this value exactly as follows:</p> <pre>SRV_STKSIZE = 48;</pre>

SRV_TMOUTINT	This specifies the interval in seconds that the timeout server will check for timed-out lock requests or hung protocol driver sessions. Values between 1 and 10 are valid; the default is 10 seconds. SRV_TMOUTINT = 10;
SYS_INGRES	This specifies the gateway data set prefix for installation files. The default value is: SYS_INGRES = 'EDBC.V2R3';

DBNMEDBC Logical Symbol Member

This logical symbol member is used to define the associations between users and database area and sort work data set names. During the connection request by a user, that user is allocated a specific database area data set for use. This database area contains the system catalogs and VANTAGE gateway mapping information. A pre-allocated sort work data set is also allocated in association with that area.

There are only two types of logical symbol specified in this member. These are of the form:

```
ING_AREA_UUUUUUU = DATABASE.AREANAME;
```

```
ING_SORT_UUUUUUU = SORT.DATASETNAME;
```

where UUUUUUU is the remote user's name in capital letters, and where DATABASE.AREANAME is the OS/390 data set name for the database area that will be associated with user UUUUUUU, and SORT.DATASETNAME is the OS/390 data set name for the sort work data set that is associated with that area.

Whenever a new user is added to the non-relational gateway, a new pair of logical symbols must be added to this member. For example, if user EDCUSR1 is to be associated with database area EDBC.V2R3.INGAREA.DEMO, then the logical symbols added to this member would be as follows:

```
ING_AREA_EDCUSR1 = EDBC.V2R3.INGAREA.DEMO;
```

```
ING_SORT_EDCUSR1 = EDBC.V2R3.INGSORT.DEMO;
```

Note: Users can share the database AREA and SORT work data sets.

The DBNMEDBC logical member is only used when accessing the VANTAGE gateway in server mode (that is, ING_MODE=SERVER;). If access is desired in Stand Alone Back End mode (that is, ING_MODE=NOSERVER;), the SABExxxx member is used.

SABExxxx or Unnnnnnn Logical Symbol Member

This member is created for each unique database area that is to be formatted and accessed. This member is concatenated after the ISVREDBC (or equivalent) member and specifies access to a specific area. You must use this member when running the createpr utility. It is also required when accessing the non-relational gateway from TSO or as a batch job.

II_LOCAL_TCPIP	This specifies the name of your TCP/IP address space. An example is: <code>II_LOCAL_TCPIP = TCPIPSTC;</code>
II_DBMS_SERVER	This parameter is required when accessing the non-relational gateway in single-user mode. It contains the OS/390 subsystem name. An example is: <code>II_DBMS_SERVER = EDBC;</code>
ING_MODE	This parameter specifies that the gateway is to run in single user mode. It must be entered exactly as follows: <code>ING_MODE = NOSERVER;</code>

Optional Logical Symbols

There are a number of logical symbols that are not normally used during the operation of the gateway. It is possible to specify these logicals to modify the behavior of the gateway. This section describes some of these logicals.

II_GCCI1_LOGLVL	This symbol specifies the level of logging to be performed. The value that is specified determines what is placed in the public log. The public log is identified by the logical symbol II_GCCI1_LOG.
-----------------	---

The values that can be specified are 0, 4 and 6. Their meanings are:

- 0: Silent (log nothing)
- 4: Log the following:
 - GCC START/STOP status messages
 - Fatal GCC errors that cause the GCC process to stop
 - Connection-specific errors (that break a specific connection).
- 6: Same as 4, but also log connection setup and termination messages for the application layer and the transport layer.

This value will default to the equivalent specification. An example is:

```
II_GCCI1_LOGLVL = 4;
```

II_GCCI1_ERRLVL This symbol specifies the level of error logging to be performed. The value given is a threshold. If a value of 0 is given, then no error logging will be done. If a value of 3 is given, then error messages with assigned levels of 0 through 3 will be logged. This value will default to the equivalent specification. An example is:

```
II_GCCI1_ERRLVL = 4;
```

IIPARM Clist Description

The IIPARM Clist works in conjunction with the logical symbol library to configure the correct logical symbols to access the gateway. This Clist is used by the IGWFDBA utilities. This Clist allocates the following files, which are necessary for the execution of the gateway under TSO:

- IIPARM – EDBC server logical symbol file, which consists of one or more concatenated data sets
- IIERLOG – Net messages
- IIVDBLOG – verifydb message log
- IIGWTR – gateway messages
- PRINTQRY – contains all SQL queries when the command set printqry is issued

Description of IIPARM Parameters

The permissible parameters are used to dynamically configure the allocation sequence of the IIPARM file. The acceptable parameters are as follows:

Parameter	Description
ISVR()	Sets default server access symbol member. If not null, it will be appended to ISVR and concatenated to IIPARM allocation.
DBNM()	Sets database location symbols. If not null, it will be appended to DBNM and concatenated to IIPARM allocation.

Parameter	Description
ERLOG()	Specifies the IIERLOG allocation status. Can have one of three values: <ul style="list-style-type: none"> ERLOG (DUMMY) – Specifies that IIERLOG will be allocated as a dummy data set. This is the default value. ERLOG (*) – Specifies that IIERLOG will be allocated to the TSO terminal.
ERLOG(dsname)	Specifies that IIERLOG will be allocated to a predefined data set. If name is not in quotes, then TSO will prefix the name with the value of the TSO user prefix.
VDBLOG()	Specifies the IIVDBLOG allocation status. Can have one of three values: <ul style="list-style-type: none"> VDBLOG (dummy) – Specifies that IIVDBLOG be allocated as a dummy data set. This is the default value. VDBLOG (*) – Specifies that IIVDBLOG be allocated to the TSO terminal. VDBLOG (dsname) – Specifies that IIVDBLOG be allocated to a predefined data set. If dsname is not in quotes, then TSO will prefix the name with the value of the TSO user prefix.
SABE	Sets logical symbols to access a specific gateway database area and sort work data set. The symbol name has this form: SABEnnnn
USER	Sets logical symbols to access a specific gateway database area and sort work data set. The symbol name has this form: Unnnnnnn
FREE	Frees all previously allocated files by filename.
HELP	Shows this output.
DEBUG	Shows internal execution of IIPARM Clist.
INSRV()	Sets the EDBC server access symbol member. This parameter is similar to the ISVR(). Unlike ISVR(), this parameter allows an explicit member that contains the server initialization parameters to be specified. It is primarily used by the IGWFDBA utility.

Parameter	Description
AREA()	Sets the gateway area symbol member. This parameter is similar to the USER () and SABE () parameters. However, unlike the other parameters, this parameter allows the full member name to be specified. It is primarily used by the IGWFDBA utility.
SILENT	Suppresses messages displayed when the IIPARM Clist executes. When this parameter is specified, no notification messages are displayed. It is primarily used by the IGWFDBA utility.
STOGRP()	Specifies that the Storage Group Catalog data set is to be allocated as the IISTOGRP ddname. The allowed values are: <ul style="list-style-type: none"> ■ YES – Allocates the IISTOGRP and IISTOLOG catalogs. If this value is specified, then the IISTOGRP ddname is allocated using the default data set name EDBC.V2R3.FILES.IISTOGRP and the IISTOLOG ddname is allocated using the default data set name EDBC.V2R3.FILES.IISTOLOG. ■ NO – Allocates the IISTOGRP and IISTOLOG catalogs as dummy data sets.

IIPARM Customization

The IIPARM Clist is customized during the IIVP stage2 jobstream execution. The ISVR (), PREFIX (), and STOGRP () parameters have defaults. The resulting Clist is placed in EDBC.V2R3.FILES.CLIST.

This data set is a fixed-block data set with an LRECL of 80. You should be able to access the IIPARM member with the SYSPROC ddname. If so, the defaults are consistent with the installation-specific naming conventions.

Examples of IIPARM Use

After you have copied this Clist into the production Clist libraries and customized it, you can use it to allocate the logical symbol file. The HELP option describes its functions. The following provides two examples of how this Clist is used. This discussion assumes that the logical symbol file has been configured and the Clist is being accessed from TSO.

Example 1

Set up the environment to access the gateway Comm server using the following command:

```
%iiparm isvr(edbc)
```

This allocates the IIPARM file to provide access to the Net Comm server that was started using the parameters in the ISVREDBC member in the logical symbol file.

The following messages will appear on the TSO console:

```
**IIPARM** IIPARM DSN = 'EDBC.V2R3.FILES.IIPARM'  
**IIPARM** IIPARM MBR = 'ISVREDBC'  
**IIPARM** ALL FILES ALLOCATED SUCCESSFULLY
```

Example 2

Set up the environment to access the database areas for EDCUSR1. This step assumes that member UEDCSR1 in the logical symbol library has been customized.

```
%iiparm isvr(edbc) user(EDCSR1)
```

This allocates all files to allow access to the VANTAGE gateway from TSO. This user will use the database area that will be allocated by the gateway exclusively, for the duration of the connection.

The following messages appear on the TSO console:

```
**IIPARM** IIPARM DSN = 'EDBC.V2R3.FILES.IIPARM'  
**IIPARM** IIPARM MBR = 'ISVREDBC'  
**IIPARM** IIPARM MBR = 'UEDCSR1'  
**IIPARM** ALL FILES ALLOCATED SUCCESSFULLY
```


Customization

During stage1 of the installation, you provide site-specific information. As described in the installation procedure, this information is used to configure the IIVP stage2 jobstream. In addition, the site-specific information is used to customize several JCL members that are used to customize the EDBC server, thereby reducing the editing required before a sample JCL member can be used.

During the IIVP stage2 phase, the following data sets are customized with this site-specific information:

- EDBC.V2R3.SAMPLE.CNTL
- EDBC.V2R3.FILES.PROCLIB
- EDBC.V2R3.FILES.CLIST

This JCL can be submitted without any changes, but it is advisable to review it before execution.

The tables below list the JCL and its functions. The IINAMEI1, STARTI1, and EDBC11 JCL are listed as they would be named with the default installation code, I1.

EDBC.V2R3.SAMPLE.CNTL

The SAMPLE.CNTL partitioned data set contains JCL that can be used to customize various components of the EDBC server:

JCL Title	Function	Description
ASMACF2	CA-ACF2 exit	<p>Customizes the CA-ACF2 security exit, IIACF. Compiles the source member IIACF2 in the data set with the default name:</p> <p>EDBC.V2R3.FILES.ASM</p> <p>This JCL is required only if the CA-ACF2 security exit requires modification or if the IIACF load module is not present in the EDBC.V2R3.BACK.LOAD library.</p>

JCL Title	Function	Description
ASMPSERV	IIPSERV module	Re-links the IIPSERV load module that contains the protocol server initialization parameters for all installed protocol servers. You do not need to execute this JCL after the IIVP stage2 has executed successfully. Use this JCL to make changes, if required.
ASMRACF	RACF exit	Customizes the RACF security exit, IIRACF. Compiles the source member IIRACF in the data set with the default name: EDBC.V2R3.FILES.ASM This JCL is required only if the RACF security exit is modified.
ASMRCALL	Data Archive Retrieval exit	Customizes the Archive Retrieval exit. Compiles and link edits the source member IIRECALL in the data set with the default name:
ASMRMODE	Sample VTAM SNA logmode table	Contains VTAM MODEENT statements defining the VTAM mode tables if SNA_LU0 or SNA_LU62 protocols have been selected.
ASMTSS	CA-TSS exit	Customizes the CA-TSS security exit, IITSS. Compiles the source member IITSS in the data set with the default name: EDBC.V2R3.FILES.ASM This JCL is required only if the CA-TSS security exit is modified.
IGWFCOPY	Copy IGWF modules from BACK.LOAD to LINKLIST	JCL to copy IGWF subsystem modules into a LINKLIST library.
IGWFXSSD	Initializes the IGWF subsystem	JCL to dynamically initialize the IGWF subsystem.
IGWFZAP	Refresh and/or map IGWF subsystem modules	JCL to dynamically refresh and/or map IGWF subsystem modules.
IIARCHIV	HSM sample recall exit	Tests the IIRECALL exit without bringing up a EDBC server. This job invokes the IIRECALL exit, produces a report, and then shuts down.
IICLEAN	Recover from a RC=20 abend of the EDBC server	This JCL should be submitted when the EDBC server terminates with a RC=20 during server installation.
IIGTFSQL	Read GTF trace	Invokes the IIGTFSQL utility.

JCL Title	Function	Description
IINAMEI1	Name server file allocation JCL	<p>Can be used to explicitly allocate the name server files needed by the EDBC server. The files that this jobstream allocates are listed below with their default names:</p> <p>EDBC.V2R3.FILES.NAME.IICOMSRV.I1</p> <p>EDBC.V2R3.FILES.NAME.IIDB2.I1</p> <p>EDBC.V2R3.FILES.NAME.IIIMS.I1</p> <p>EDBC.V2R3.FILES.NAME.IIINGRES.I1</p> <p>EDBC.V2R3.FILES.NAME.IILOGIN.I1</p> <p>EDBC.V2R3.FILES.NAME.IINODE.I1</p> <p>EDBC.V2R3.FILES.NAME.IIVSAM.I1</p> <p>EDBC.V2R3.FILES.NAME.IIVANT.I1</p> <p>If you execute this JCL, you must recreate any definitions entered in the netu utility. The iiname utility does not need to be run.</p>
EDBCEDBC	Server batch JCL	Starts the server as a batch job using in-stream JCL.
RTIAPPL	Sample VTAM SNA APPL definition	Contains the VTAM APPL statements defining the VTAM application node names if SNA_LU0 or SNA_LU62 protocols are selected.
STARTI1	EDBC server batch JCL	Allows the EDBC server to be submitted as a batch job. Assumes that the EDBCI1 member in the gateway PROCLIB has been copied to an installation user PROCLIB.
UPD01PSB catalog	Update the IIGW01PSB catalog	JCL to update the VANTAGE partition processing options.

EDBC.V2R3.FILES.PROCLIB

The EDBC.V2R3.FILES.PROCLIB data set contains procedures that are used by OS/390 TSO or batch users of the EDBC server:

JCL title	Function	Description
EDBCI1	Customization EDBC server cataloged procedure	Used to start the EDBC server as a started task or batch job. Should be copied to an installation user proclib.

Accessing Vantage Data

There are a number of ODBC compliant tools available to you on a number of operating systems, including Microsoft Windows and various UNIX platforms. Before you can access data in EDBC through an ODBC interface, you must install the EDBC Client software on the platform where the ODBC application will run. The EDBC Client implements a specific ODBC *driver* required by the architecture in accessing an ODBC *data source*.

Follow the instructions for installing the EDBC Client on the specific platform you will be using. In this guide, all EDBC Client discussions and examples are for the Microsoft Windows platform.

Defining EDBC Server to EDBC Client

Before you can access Vantage through ODBC or JDBC, you need to define your EDBC Server to EDBC Client using the EDBC Network Utility.

To begin this process, start the EDBC Network Utility. The first time this utility is invoked, there are no EDBC nodes defined to the client. An EDBC *node* is a definition that points to a specific instance of EDBC Server running on a mainframe system.

To define a node to the EDBC client:

1. Click the Add Virtual Node icon.

Alternatively, right-click the Node tree element and select Add.

The EDBC Network Utility displays an Add Virtual Node Definition dialog where you define your new EDBC Server node definition.

2. Fill in the fields in this dialog using the EDBC information shown in the following figure:

The screenshot shows a dialog box titled "Add Virtual Node Definition". It has a close button in the top right corner. The dialog is divided into three sections:

- Virtual Node:** A text field containing "TS04". To the right are "OK" and "Cancel" buttons.
- Login Information:** A section with a "User Name" field containing "bruti02", a "Password" field with asterisks, and a "Confirm Password" field with asterisks. To the right is a checked "Private" checkbox.
- Connection Information:** A section with a "Node" field containing "TS04.ca.com", a "Protocol" dropdown menu set to "wintcp", and a "Listen Address" field containing "134". To the right is an unchecked "Private" checkbox.

3. Test your new node definition by right-clicking on that node in the tree and selecting the Test Node option.

If you have correctly defined the new node, and if that node (EDBC/Server) is active, a pop-up appears, indicating that the node test was successful. If the node definition is not correct, or the target EDBC/Server is not running – or not currently accessible due to a transient network problem – a pop-up indicates that the test was not successful.

An ODBC Example: EDBC Client for Windows

The EDBC Client includes an ODBC display application that may be used to access Vantage data on the mainframe. Once the EDBC Client is installed, this display application is available.

When you have established a node definition connection to an EDBC/Server instance on a mainframe, you can view Vantage data – assuming Vantage is running and available to the EDBC/Server.

Expand the tree associated with the tested EDBC node to display two branches of the tree (Gateways and Advanced Node Parameters). When you expand Gateways, the EDBC Client connects to the EDBC/Server and a list of available EDBC gateways installed in your EDBC/Server configuration appears. EDBC/Server supports various gateways, but Vantage data is only accessible through the VSAM gateway. The VSAM gateway appears under the Gateways branch of the EDBC Network Utility tree.

To view Vantage data:

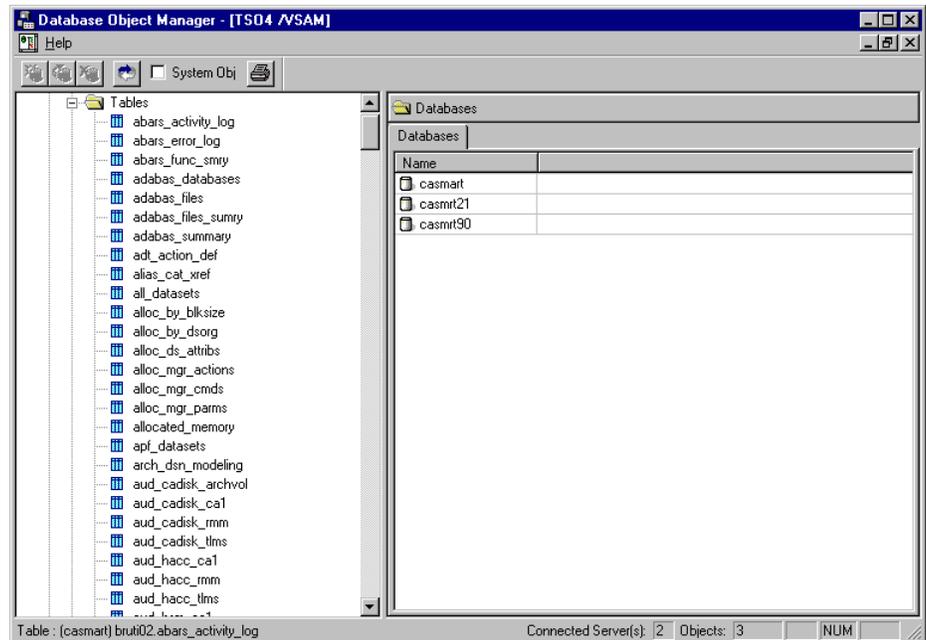
1. Click the VSAM icon to highlight the VSAM gateway.
2. Click the Database Object Manager icon on the toolbar to start the Database Object Manager.

Alternatively, right-click the VSAM icon and select Database Object Manager.

The Database Object Manager utility starts and connects you to EDBC/Server to get a list of EDBC *partitions* defined to that EDBC/Server. This is a list of the partitions you have defined to that EDBC/Server, which may be one or more Vantage subsystems and/or one or more actual VSAM-based EDBC partitions.

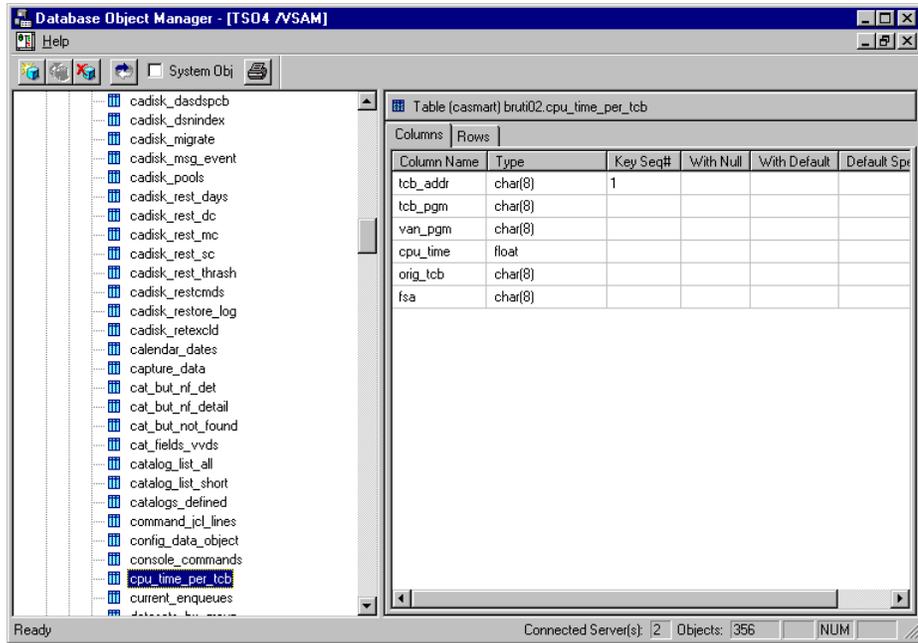
To get a list of the Vantage objects you have successfully defined to EDBC:

1. Expand the tree associated with the desired Vantage partition to view three branches: Tables, Views, and Schemas.
2. Expand the Tables branch to view the Database Object Manager and a list of the tables defined to the partition. See the following figure:

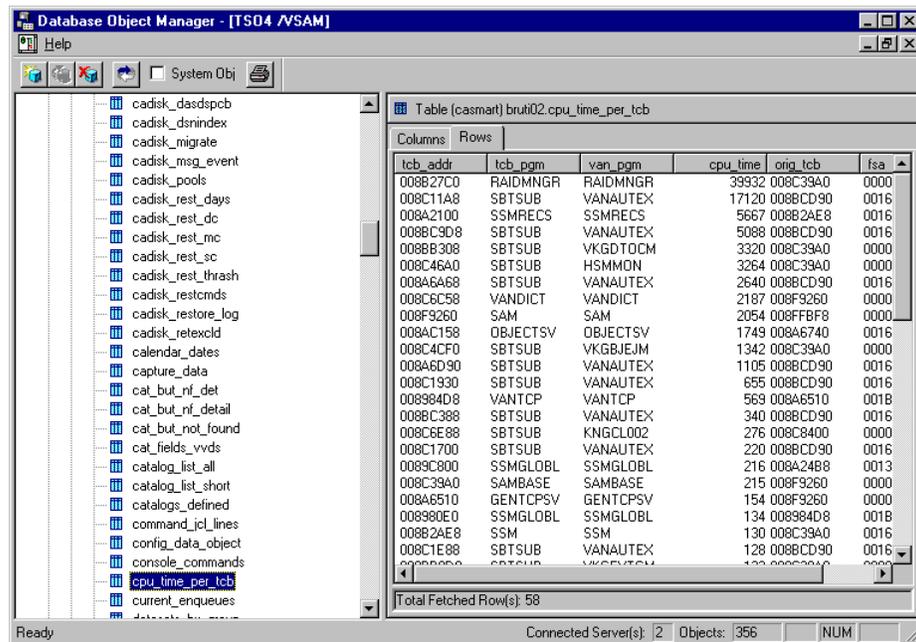


3. Click any of these EDBC tables (Vantage objects), and the Database Object Manager retrieves a mapping of the fields (columns) available through that table. Tabs across the top of the window show the field definitions.

- Click the Rows tab to get the actual Vantage data associated with this table and display that data as rows of a database table. See the following figure:



You have now accessed Vantage data through ODBC using the EDBC Client software Database Object Manager. See the following figure:



ODBC Data Sources in Microsoft Windows

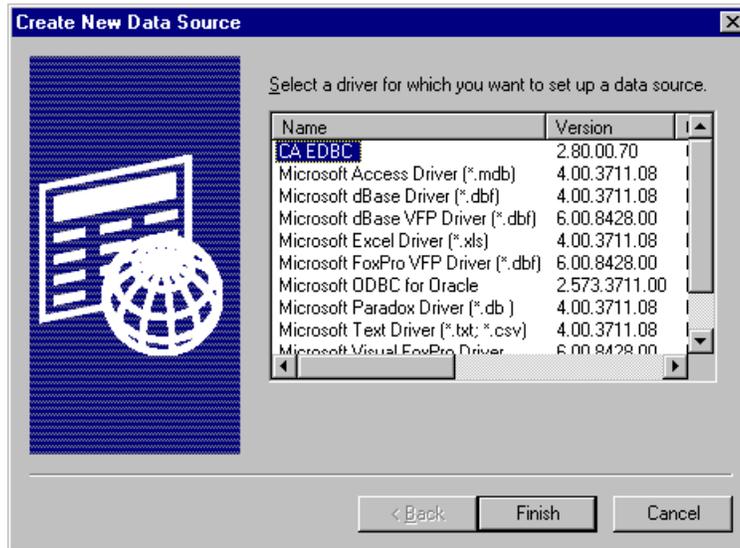
The EDBC Client Database Object Manager takes a shortcut through EDBC Client that does not require an ODBC data source normally required by ODBC applications. On Windows platforms, Microsoft provides an ODBC Data Sources administrative program you can use to define and manage actual ODBC data sources.

Before you can use ODBC compliant applications, you will need to define an ODBC data source that points to a specific EDBC node as defined using procedures outlined below. See Microsoft documentation on the use of the ODBC Data Sources facility for more information.

To add a new ODBC data source:

1. Start the Microsoft ODBC Data Sources program.
2. Select the SYSTEM DSN tab.
3. Click Add to start the process of adding a new ODBC data source.

You will see a list of ODBC drivers installed on your Windows platform, including the CA EDBC driver.

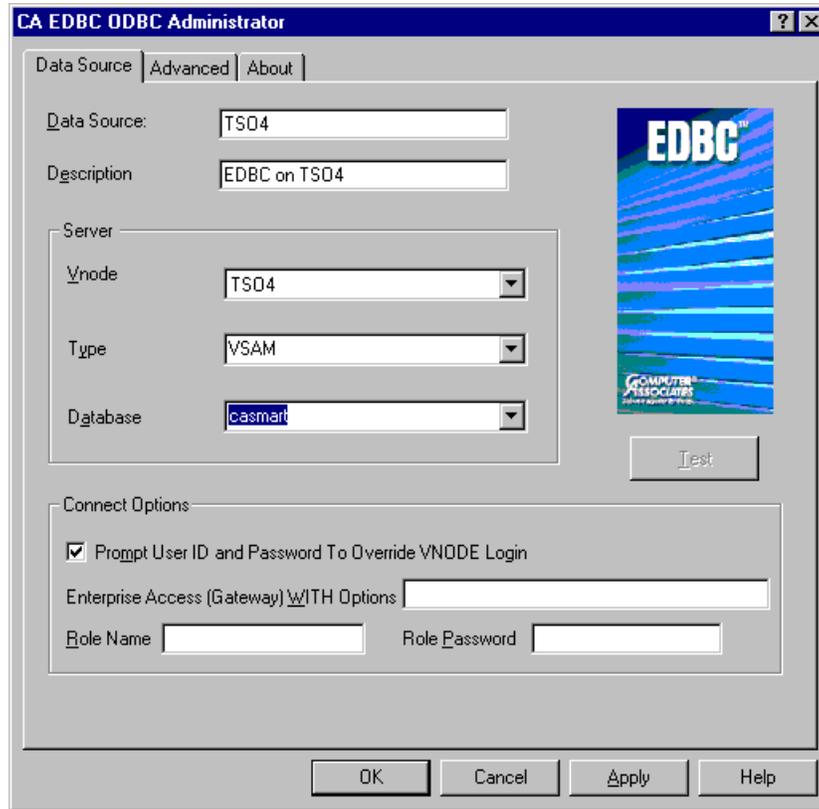


4. Click CA EDBC driver, and click Finish.

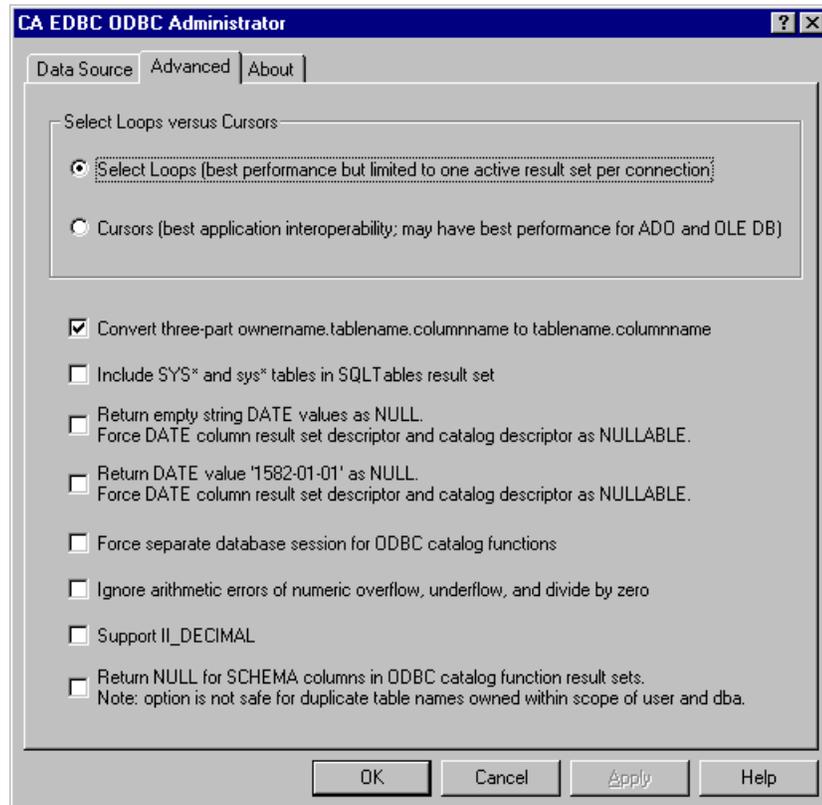
A CA EDBC ODBC Administrator dialog appears in which you define your new ODBC data source.

5. Type a meaningful name and description for your ODBC data source in the fields provided.

6. Select an EDBC node and database to be associated with the ODBC data source. Note that database type defaults to VSAM.



7. Select the Advanced tab and make sure that the following options are checked:
 - Select Loops
 - Convert three-part ownername.tablename.columnname to tablename.columnname



8. Return to the Data Source tab and click Test to test the ODBC data source prior to saving your changes.
9. Click Apply or OK when finished.

The new ODBC data source is defined and is now available to ODBC applications. (Close the Microsoft ODBC Data Sources application when you are finished.)

Example: Microsoft Excel

Once you have added an EDBC ODBC data source, you can use any ODBC compliant application to access Vantage data. For example, you might use Microsoft's Excel spreadsheet application.

Accessing an ODBC Data Source

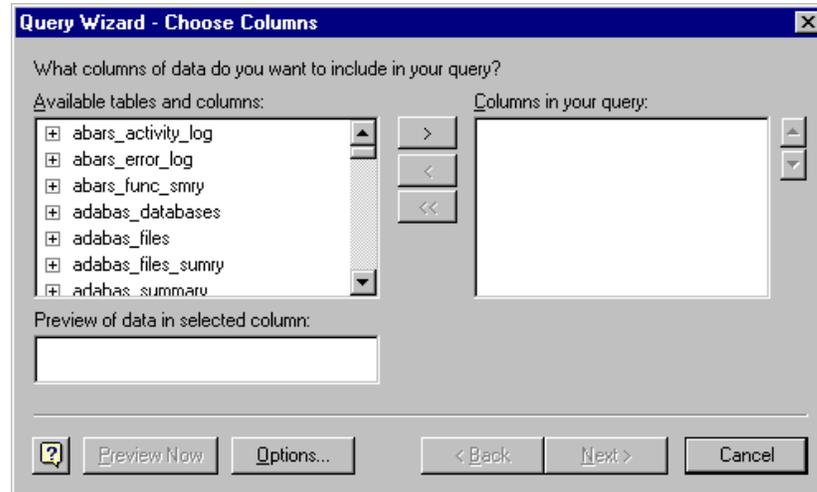
To access an ODBC data source:

1. Start Microsoft Excel.
2. Select Data, then Get External Data, then New Database Query.

Note that Excel requires an ODBC component of Microsoft Office be installed; if it is not installed on your machine, you may be prompted to install it before you can continue.

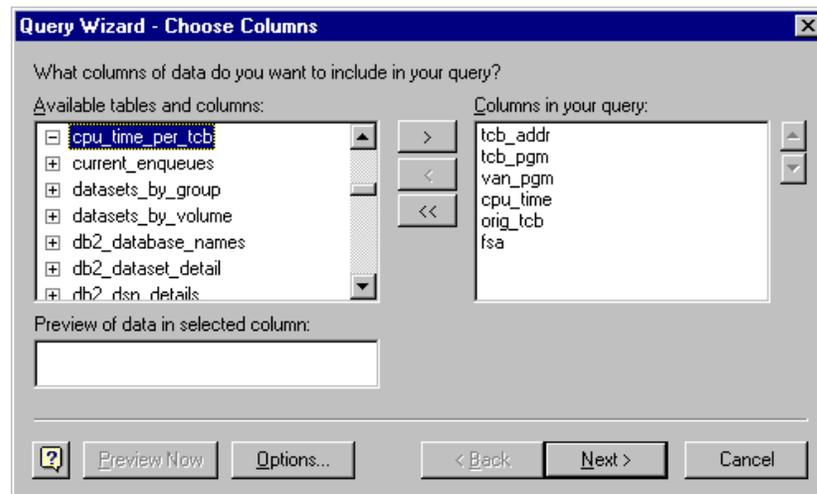
The Microsoft Query Choose Data Source dialog appears. Scroll through the list and find an ODBC data source for EDBC previously defined to Windows. You may be prompted for a User ID and Password, depending on how the ODBC data source and EDBC node was defined. In this dialog, the only fields you need to fill in are User ID and Password.

Microsoft Query now connects to the EDBC Server and displays the Choose Columns dialog, which contains a list of EDBC tables:



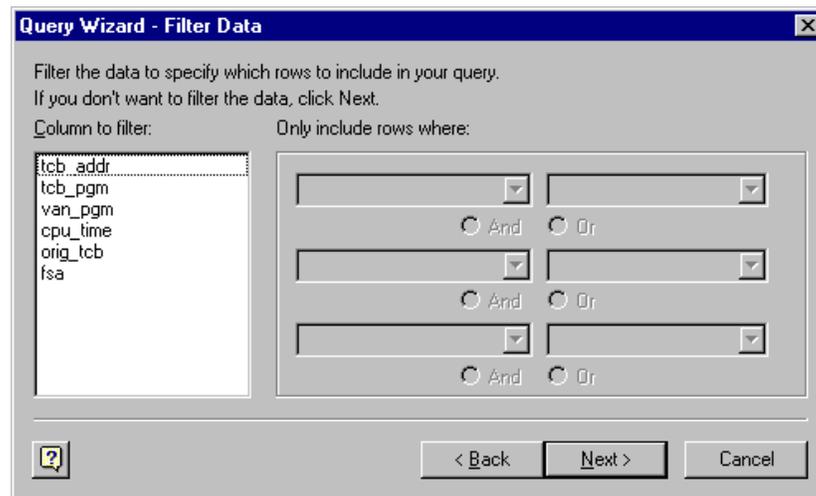
Selecting Tables

You can select an entire table by clicking the table name and clicking the > button; or select one or more fields (columns) from the table by expanding the table definition (click the plus sign). Then select one field at a time and click the > button.



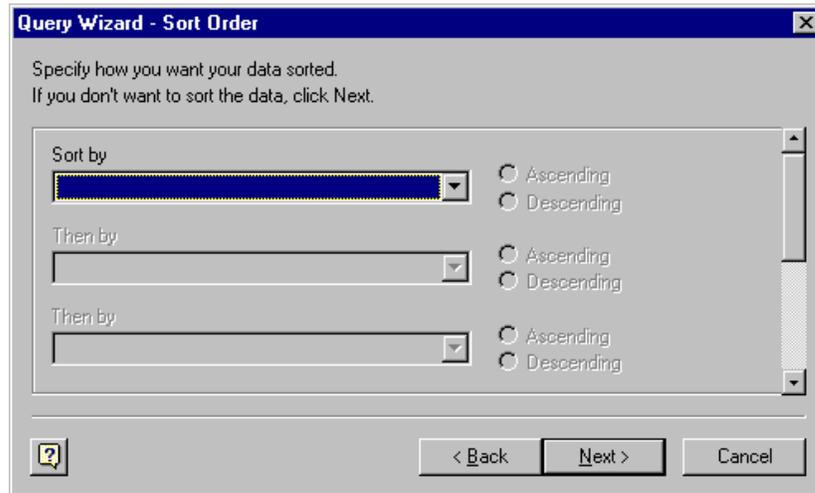
Define Filter Criteria

After you select a table or specific columns of a table to be returned to Excel, click Next. The Filter Data dialog appears, allowing you to define filter criteria for selecting Vantage data based upon logical expressions involving the content of one or more of the fields in the object:

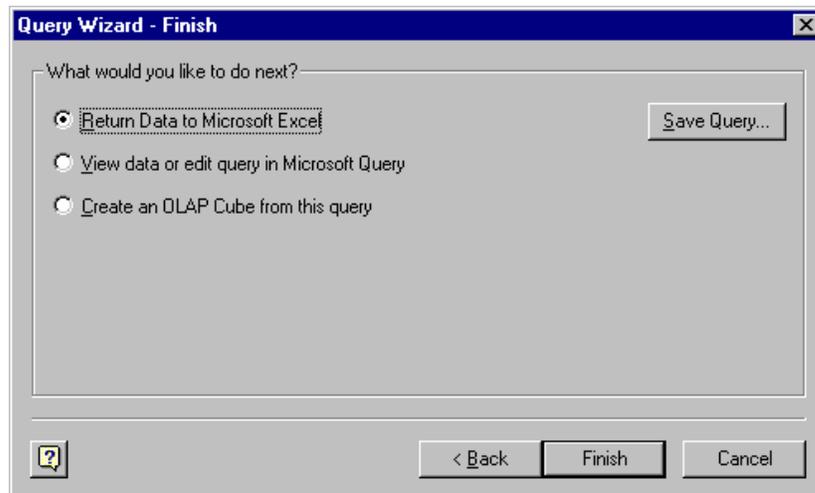


Define Sort Criteria

Once you have entered any filter criteria, click Next. The Sort Order dialog appears, allowing you to define criteria for sorting the resulting rows of data as you want them to appear in your Excel spreadsheet:



Click Next. The Finish dialog appears. Indicate what you would like to do with the ODBC query results. The default is to return the data to Microsoft Excel. Click Finish to begin the process of getting data from Vantage.



The Returning External Data to Microsoft Excel window appears. See the Microsoft Excel documentation for how to use this window and Microsoft Query.

Displaying Vantage Data

For this example, accept the defaults and return the data to the existing Excel worksheet and the default rows and columns. The data you have requested is returned to Excel in spreadsheet format. You can use various facilities in Excel, such as its graphs and charts features, to massage and display this data in any way you wish.

	A	B	C	D	E	F
1	tcb_addr	tcb_pgm	van_pgm	cpu_time	orig_tcb	fsa
2	008B27C0	RAIDMNGR	RAIDMNGR	43661	008C39A0	00006B70
3	008C11A8	SBTSUB	VANAUTEX	18965	008BCD90	001659C8
4	008A2100	SSMRECS	SSMRECS	7611	008B2AE8	00165800
5	008BC9D8	SBTSUB	VANAUTEX	5089	008BCD90	00165DF8
6	008C46A0	SBTSUB	HSMMON	3666	008C39A0	000064F8
7	008BB308	SBTSUB	VKGDTOCM	3320	008C39A0	00006218
8	008A6A68	SBTSUB	VANAUTEX	2640	008BCD90	001658D8
9	008C6C58	VANDICT	VANDICT	2187	008F9260	00006A00
10	008F9260	SAM	SAM	2054	008FFBF8	00006F88
11	008AC158	OBJECTSV	OBJECTSV	1749	008A6740	00165100
12	008C4CF0	SBTSUB	VKGBJEJM	1503	008C39A0	00006458
13	008A6D90	SBTSUB	VANAUTEX	1105	008BCD90	00165A18
14	008C1930	SBTSUB	VANAUTEX	655	008BCD90	00165BA8
15	008984D8	VANTCP	VANTCP	569	008A6510	001B3300
16	008BC388	SBTSUB	VANAUTEX	346	008BCD90	00165AB8
17	008C6E88	SBTSUB	KNGCL002	298	008C8400	00006C08
18	008C1700	SBTSUB	VANAUTEX	220	008BCD90	00165A68
19	008C39A0	SAMBASE	SAMBASE	215	008F9260	00006670
20	0089C800	SSMGLOBL	SSMGLOBL	196	008A24B8	0013E710
21	008A6510	GENTCPSV	GENTCPSV	163	008F9260	000062D0
22	008980E0	SSMGLOBL	SSMGLOBL	134	008984D8	001B32B0
23	008B2AE8	SSM	SSM	130	008C39A0	00165FB0

Example: Web with FrontPage

Using the Vantage ODBC interface, you can access and display Vantage data through the development of web applications. A number of web development tools with built-in ODBC wizards make it possible to produce a simple web application without the need to write a single line of code. One such tool is Microsoft FrontPage.

To create a simple web application using FrontPage, start FrontPage. A blank web page appears that defaults to an HTML document type.

Changing Document Type

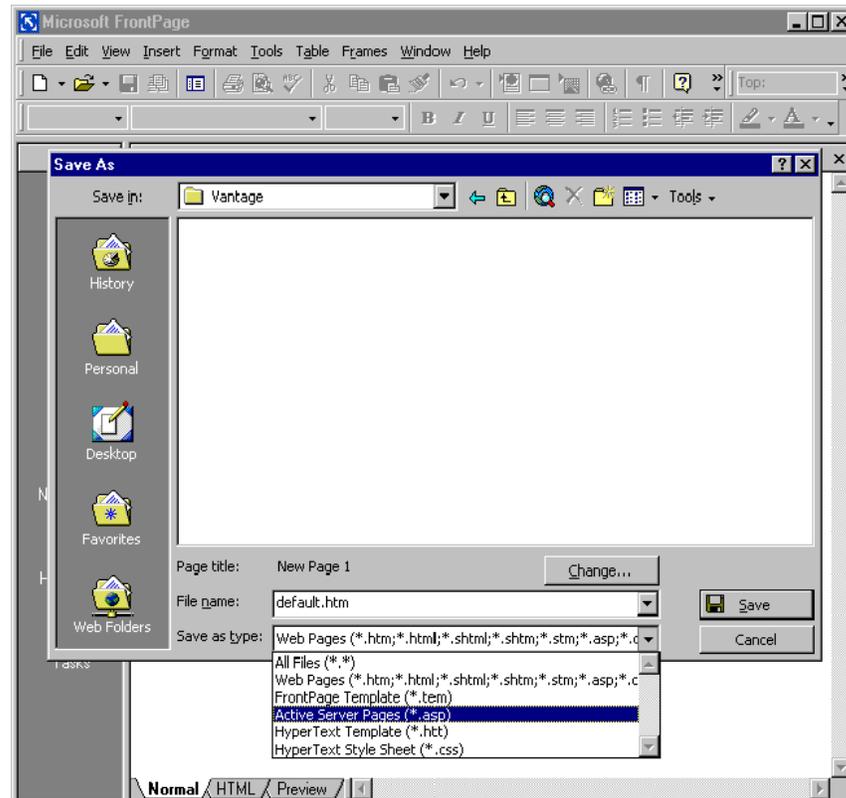
Before you can use the ODBC wizard in FrontPage, you must change the document type from HTML to ASP. FrontPage only allows you to use the ODBC wizard for ASP web pages. Therefore, you need a Web Server running the Microsoft Internet Information Server (IIS) to host your application.

To change the initial document type from HTML to ASP, choose Save As from the File menu.

By default, FrontPage chooses the My Webs subfolder in the Personal folder associated with your Windows desktop. You may want to create a folder within the My Webs folder for your new web application, or you may want to select another folder associated with an existing IIS web environment.

Provide a meaningful name for the web page you are about to save. Note that IIS considers a page named default.htm or default.asp to be the primary web page for a web application. When defining a web application to an IIS server, you can, however, rename the primary (default) web page.

Once you have named your new web page, choose Save As Type and the Active Server Pages (*.asp) option. (See the following figure.) Click Save. You have changed the document type from HTML to ASP. You may begin building the web page.



The purpose of this example is to show you how to use the FrontPage ODBC wizard to gain access to Vantage data. Therefore, our web page will contain only what FrontPage calls a *database results object*.

Note: FrontPage provides various features allowing you to add banners, images, and descriptive text to a web application.

If you have successfully followed the procedures for converting this initial page from HTML to ASP format, you can choose Database from the Insert menu. If the Database option is not available to you, the document may not have been successfully converted to ASP format. In this event, carefully repeat the procedures for converting to ASP format.

If you still cannot access the Database option, you may not have installed the ODBC component of Microsoft Office, which is required for creating web pages that need ODBC controls using FrontPage.

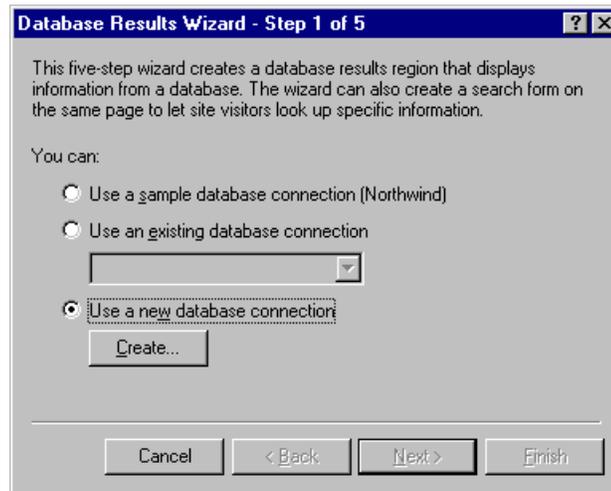
Creating Web Page

To start the ODBC Database Results Wizard:

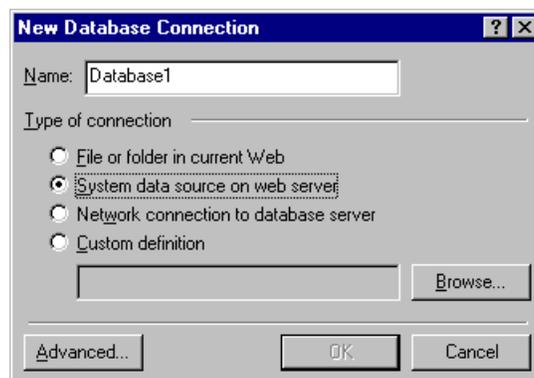
1. Choose Database from the Insert menu, and then choose Results.

The Database Results Wizard asks you to identify the ODBC data source to be used to extract and display information on the web page. The first time you perform this function in FrontPage, there are no ODBC data sources listed under Use an Existing Database Connection.

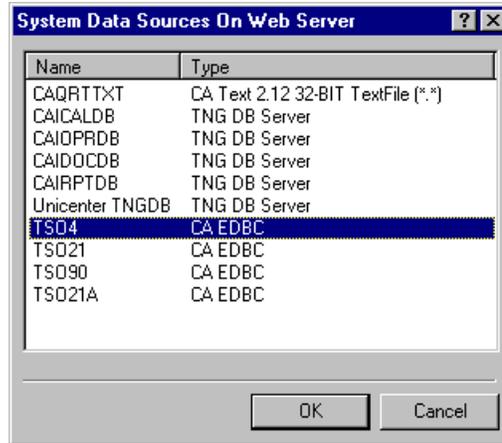
2. Select Use a New Database Connection, and click Create.



3. In the dialog that appears, click Add to access the New Database Connection dialog.
4. In the New Database Connection dialog, select System Data Source On Web Server, and click Browse.



A list of ODBC data sources defined to your server appears along with the EDBC data sources you may have defined before building your web application.



5. Select the ODBC data source you want to use in creating your web page, and click OK.

You are returned to the New Database Connection dialog.

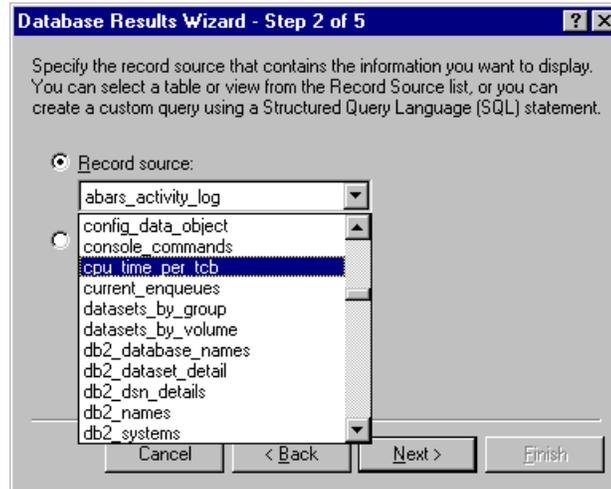
6. Click Advanced and enter a User ID and Password to be used in connecting to the mainframe associated with the ODBC data source you have selected.
7. Click OK to save this information.

Note: Before closing the New Database Connection dialog, you may want to change the default name of this connection.

8. To save your new database connection information, click OK.
9. Continue to click OK until you have returned to the Database Results Wizard.

The ODBC data source you have just defined is now selected as the database connection to be used in creating your web page.

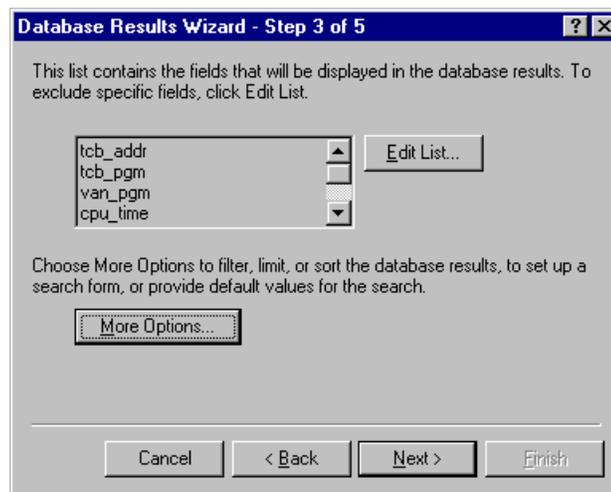
10. Click Next to go to Step 2 of the Database Results Wizard procedure:



The wizard now connects to your mainframe through this ODBC connection to get a list of ODBC database tables available through this data source. In this example, the wizard creates the SQL query. (You could select Custom query and click Edit to create your own SQL command.)

The same information that was in the Excel example in the “Example: Microsoft Excel” appendix is displayed. Scroll to the `cpu_time_per_tcb` table and select this table.

12. Click Next to go to Step 3 of the Database Wizard procedure:

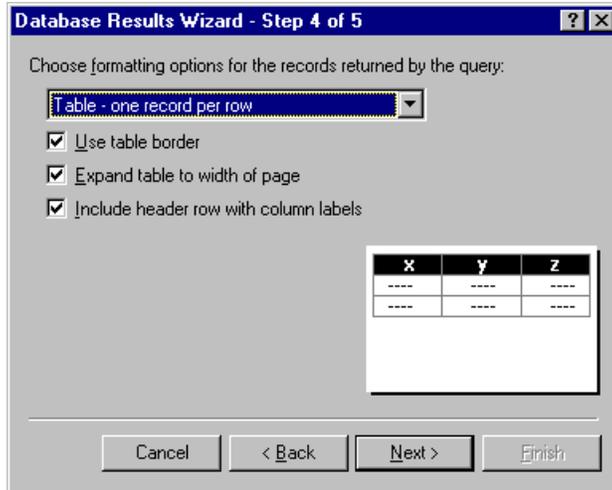


The wizard again connects to your mainframe to get all of the field definitions associated with this ODBC table. By default, all of the fields are selected for display on the resulting web page.

To remove one or more of these fields from your web page or to change the order in which they appear, click Edit.

In the wizard, you can define any filters and identify how the resulting data should be sorted by clicking More Options. Also, you can change the limit for how many database records from your ODBC data source you want displayed in your web application (default is 256).

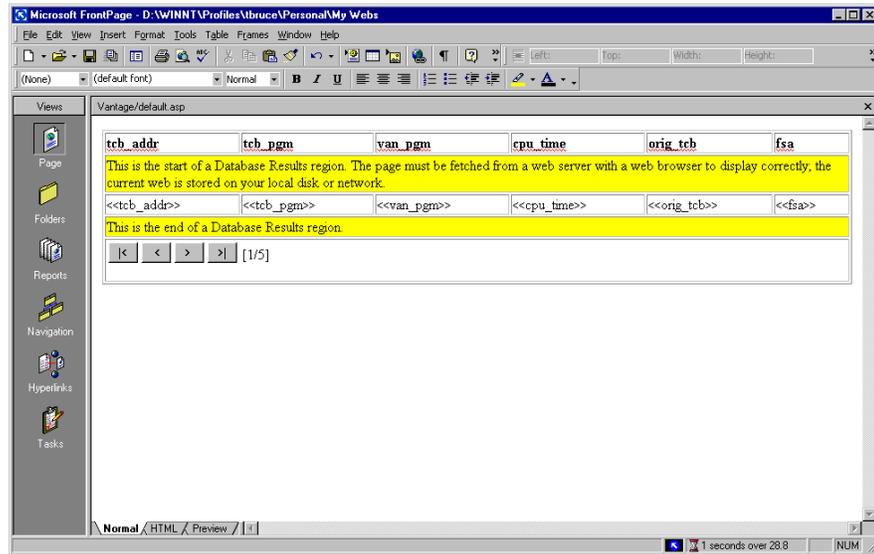
13. Define the fields you want to display and set any other options relating to record selection and sorting.
14. Click Next to go to Step 4 of the Database Wizard procedure:



You may now identify how you want your database records to be displayed. Choose the display format that best fits your needs.

15. Click Next to move to Step 5 of the Database Wizard procedure.
16. Select the final formatting options and then click Finish.

You have now created a simple web page capable of getting data from a mainframe system running the Vantage product. In FrontPage, you will see a descriptive table placeholder on your web page; you will not see the mainframe data. You may develop your web page around this placeholder to add other text and images.



You can save this web page “as is” and make the web application available through an IIS Web Server.

This example only demonstrates how to use the FrontPage Database Wizard to generate the web page and underlying ADO code for you.

You do not need to use the FrontPage Database Wizard to access ODBC data. You may use FrontPage to create and format your web page, building your own ADO code to extract information from Vantage using ODBC. This may be desirable, for example, when you need to display information in a way that is not easily generated by the wizard.

Installing Multiple Gateways

The EDBC installation procedure supports the installing of multiple gateways in a single stage1/stage2 run. To install multiple gateways perform the following:

1. Download the files from the product tape as outlined in "Installing the Gateway" chapter.
2. Edit member IGWFSTGA in library EDBC.V2R3.FILES.ASM and make the following changes:
 - a. The stage1 input statements for all the gateways are included in IGWFSTGA. Delete or comment out the statements for the gateways which are **not** being installed.
 - b. Customize the stage1 input for the remaining gateway entries (see the Customizing the Stage1 Input section of the "Installing the Gateway" chapter of the appropriate *Advantage EDBC Installation and Operations Guide*).
 - c. Modify the PRODUCTS= operand of the IGWFBLD statement. Delete the product codes for gateways that are **not** being installed.
3. Run stage1.
Customize and submit member IGWFSTGA of library EDBC.V2R3.SAMPLE.CNTL.
4. Run stage2.
Submit jobs in library EDBC.V2R3.SAMPLE.CNTL (see the Submitting the Stage2 Jobstream Jobs section of the "Installing the Gateway" chapter of the appropriate *Advantage EDBC Installation and Operations Guide*).
5. Perform the final installation and verification procedures as described in the "Installing the Gateway" chapter of the appropriate *Advantage EDBC Installation and Operations Guide*.

Index

A

abend codes and messages
CCI, 7-10
SNA LU0, 7-3

ACBNAME parameter, 4-15

Access
to data sets, 8-2
to gateways, 2-5, 8-1
to multiple systems, 10-28

activate (command), 10-4

Active
drivers, 6-6
threads, 10-5

Activity display, 10-7

Address space
components, 2-3
dumping, 10-5
gateway, 2-2
server, 3-3
TSO, 2-5, 3-3, 10-31

Allocating
and loading data sets, 4-5

Alternate translation tables, 7-19

Applications, porting, 1-4

Area
backing up, 10-13
checking consistency of, 10-20
compressing, 10-16
copying source to target area, 10-17
described, 2-6
diagram, 2-8
formatting gateway, 6-27
restoring, 10-14
size, 8-2, 10-9, 10-16

space allocation, 10-9
virtual storage requirements, 3-3

Assembler data conversion, 9-8

Authorization ID, 10-32

B

Backing up files, 4-4

Batch file input, 6-11

C

CA-ACF2, 7-14, B-1

Catalogs, 8-3

CA-Top Secret Security facility (CA-TSS), 7-15, B-2

CCI
abend codes and messages, 7-10

CCI for OS/390, 7-9

Character data type, 9-10

Clients
creating RACF profile, 7-13
described, 1-5
logical symbols location, 8-3

COBOL data conversion, 9-6

columns
selecting, D-3

Commands
defined, 1-8
delimiters, 10-3

Communication
protocol requirement, 3-1
server, 2-4, 2-5

Connecting
server to server, 10-31
to gateways, 8-1, 10-13

Conventions, 1-8

Converting data types, 9-9

createpr (utility), 10-19

CREATPR processing mode, A-10

Cylinders required, 4-4

D

Data

access, 2-5
conversion, 9-9
registration, 2-6, 9-1

Data Communications Control Block, 6-8

Data set

default prefix, 4-6
linear, 9-29
prefix, specifying logical, A-4, A-12

Data sets

allocated by stage0, 4-7
allocating and loading, 4-5
archived, A-9
read/write access to, 8-2

Data types

character, 9-10
converting to EDBC, 9-9
date, 9-17
float, 9-16
integer, 9-14
mapping of, 9-6
money, 9-21
numeric, 9-12
varchar, 9-10, 9-11

Database

distributed, 1-7
rebuilding master, 10-22
utilities, 10-19

Datatypes

imaginary, 9-25

Date data type, 9-17

DBA

access to gateways, 4-33
access to sample databases, 4-11

and users, 9-30
control, 8-3, 9-28
debugging aids for, 10-50
ID, 4-11
requirements, 3-4
resources, 5-1
tasks, 8-1, 8-2
TSO logon procedure, 4-32
utilities, 9-30

dbconst (utility), 10-20

DBMS

catalogs, repairing, 10-25
utilities, 6-33

DBNMEDBC logical, A-12

Debugging, 10-50

Dependent logical unit, 7-5

destroypr (utility), 10-21

DI (utilities), 6-14

diatedmp (utility), 6-15, 10-9

dibackup (utility), 6-19, 10-11, 10-16

dicomprs (utility), 6-23

dicopyar (utility), 6-25, 10-17

didirdmp (utility), 6-17, 10-10, 10-11

DI-file analysis scripts, 10-11

direstor (utility), 6-21, 10-14

Disk storage, 3-3

display active threads (command), 10-5

Distributed databases, 1-7

DLOGMOD parameter, 4-15

document type

ASP, B-1
changing, B-1
HTML, B-1

Documentation conventions, 1-8

Dropping tables, 9-33

dump (command), 10-5

E

EDBC

- data sources, B-4
- EDBC client, C-1
- EDBC client
 - database object manager, C-5
 - MS Windows, C-2
- EDBC Network Utility, C-1
- EDBC server
 - defining to EDBC client, C-1
- Error log
 - name server, A-5
 - specifying level, A-14
 - specifying location, A-4, A-5
- Errors
 - VSAM mapping to the gateway, A-6
- ESQL application, 6-35
- Execution mode, specifying logical, A-10
- Exits, routing, 10-51
- Extents, 8-3

F

- Facility
 - main menu, 6-6
 - profile, 6-3
- filter criteria
 - defining, D-3
- finddb (utility), 10-22
- Float data type, 9-16
- Force inactivate timeout, 7-17
- Formatting a gateway area, 6-27
- Forms file, specifying logical location, A-4
- FrontPage, B-1
 - database results object, B-2

G

- gateways
 - installing multiple, F-1
- Gateways
 - access, 2-5

- described, 1-2, 2-1
- features of, 1-4
- invoking with Net, A-8
- maintaining, 10-1
- server. *See Server*
- subsystems, 2-2

- Generic Communication Facility, 6-9
- Global Communications Architecture (GCA), 1-3
- grant (statement), 9-29
- Greenwich mean time, 7-18

H

- Hardware requirements, 3-1
- help (command), 10-5

I

- IBM TCP/IP
 - specifying logical, A-8
 - specifying parameters, 4-16
- IBM TCP/IP for OS/390
 - configuration, 7-8
- IGFW
 - initialization failures, 10-51
- IGWF subsystem
 - installing, 4-24, 4-25
 - IPL for dynamic installation, 4-33
 - upgrading, 10-23
- IGWFBLD statement, 4-20
- IGWFDBA utilities
 - compiling applications, 6-35
 - creating user partition, 6-30
 - customizing IIPARM, 6-30
 - inspecting DCCB, 6-8
 - inspecting Generic Communication Facility, 6-9
 - invoking Terminal Monitor, 6-14
 - main menu, 6-1
 - maintain areas, 10-9
 - overview, 3-5, 6-1
 - profile frame, 8-4
 - running, 6-17
 - setting up user, 8-4

IGWFINET statement, 4-11
 IGWFJOB statement, 4-9
 IGWFPSVR statement, 4-15
 IGWFSTGT member, 4-22, 4-23
 IGWFUSER statement, 4-11
 IGWFFVPA0 job, 4-29
 IGWFFVPB2 job, 4-28
 IGWFFVPI0 job, 4-29
 IGWFFVSAM statement, 4-13
 igwffzap (utility), 10-23
 II_BUFFERS logical, A-3
 II_DATABASE logical, A-3
 II_DATE_CENTURY_BOUNDARY logical, 7-19, A-3
 II_DBMS_LOG logical, A-4
 II_DBMS_SERVER logical, A-13
 II_DBTMPLT logical, A-4
 II_FILES logical, A-4
 II_FORCE_TMOUTINT logical, 7-17, A-4
 II_FORMFILE logical, A-4
 II_GCC_ID logical, A-5
 II_GCCI1_ERRLVL logical, A-14
 II_GCCI1_LOG logical, A-5
 II_GCCI1_LOGLVL logical, A-13
 II_GCN_ID logical, A-5
 II_GCNI1_LCL_VNODE logical, A-6
 II_GCNI1_LOG logical, A-5
 II_GCNI1_SVR_TYPE logical, A-6
 II_GENERIC_ERROR logical, A-6
 II_HELPDIR logical, A-6
 II_INACTV_TMOUTINT logical, 7-17, 7-18, A-7
 II_INSTALLATION logical, A-7
 II_NET_LMOD logical, A-8
 II_NO_ENQ_SUBSYS logical, A-8
 II_PROTOCOL_CCI logical, A-8
 II_PROTOCOL_RESTART_COUNT logical, A-8
 II_PROTOCOL_SNA_LU0 logical, A-8
 II_PROTOCOL_SNA_LU62 logical, A-8
 II_PROTOCOL_TCP_IBM logical, A-8
 II_PROTOCOL_TCP_IBM_BUFSIZE logical, A-8
 II_PROTOCOL_TCP_KNET logical, A-9
 II_PROTOCOL_TCP_SNS logical, A-9
 II_PROTOCOL_TCP_SNS_BUFSIZE logical, A-9
 II_PSF_POOL logical, A-8
 II_QEP_SIZE logical, A-9
 II_RECALL logical, A-9
 II_SECURITY logical, 7-14, 7-15, 7-17, A-9
 II_SMFID logical, A-10
 II_TIMEZONE logical, 7-18, A-10
 II_UTEXE_DEF logical, A-10
 iidbdb master database, 10-22
 iinamu (utility), 6-32
 IIPARM
 command procedure, 6-30
 customization, A-16
 examples, A-16
 parameters, A-14
 IIRECALL exit, B-2
 IISTAGE1 member, 4-22
 IIVP
 creating additional server, 10-28
 customizing input, 4-9
 overview, 3-6, 4-1
 IIVP stage0 jobstream
 customizing, 4-6
 data sets allocated by, 4-7
 restoring, 4-5
 IIVP stage1 jobstream
 example, 4-22
 summary, 4-9
 IIVP stage2 jobstream
 listing, 4-24
 results of, 4-3
 submitting, 4-24
 inactivate (command), 10-4
 Inactive user timeout
 displaying parameter value, 10-8

forcing, 7-17
specifying logical, A-7

Independent logical unit, 7-5

ING_AREA logical, 8-1, 8-5, 8-6

ING_MODE logical, A-10, A-13

ING_SORT logical, 8-1, 8-5, 8-6

Initialization failures
diagnosing, 10-51

installation
multiple gateways, F-1

Installation
allocating and loading data sets, 4-5
CA-ACF2, 7-14
CA-Top Secret Security facility, 7-15
code, A-7
IBM TCP/IP, 7-8
KNET TCP/IP, 7-6
OS/390 subsystems required, 4-3
overview, 3-6
RACF, 7-13
requirements, 3-1
SNA LU0, 7-1
SNA LU62, 7-4
summary, 3-7, 4-1
verification, 3-6, 5-1

Integer data type, 9-14

IPL
failures after, 10-51
timing of, 4-3, 4-33

ISO Entry SQL92 compliance, 1-8

ISVR member, A-1

ISVREDBC logical symbol members, A-3

J

JCL
creating skeleton, 6-4

Jobcard parameters, 4-9

Joins, 10-45

K

KNET TCP/IP
requirements, 7-6
specifying logical, A-9
specifying parameters, 4-15

KNET TCP/IP for OS/390
support, 7-6

L

Linear Data Set (LDS), 9-29

Local
mode, 1-5

Local time zones, 7-18

LOCK_LISTS logical, A-11

LOCK_MAX logical, A-11

LOCK_RESTAB logical, A-11

LOCK_TABSIZE logical, A-11

LOG_CPINTVL logical, A-11

LOG_SBLKS logical, A-11

Logging
not provided, 10-8
specifying level, A-13

Logical
DBNMEDBC, A-12
II_BUFFERS, A-3
II_DATABASE, A-3
II_DATE_CENTURY_BOUNDARY, A-3
II_DBMS_LOG, A-4
II_DBMS_SERVER, A-13
II_DBTMPLT, A-4
II_FILES, A-4
II_FORCE_TMOUTINT, 7-17
II_FORMFILE, A-4
II_GCC_ID, A-5
II_GCCI1_ERRLVL, A-14
II_GCCI1_LOG, A-5
II_GCCI1_LOGLVL, A-13
II_GCN_ID, A-5
II_GCNI1_LCL_VNODE, A-6
II_GCNI1_LOG, A-5
II_GCNI1_SVR_TYPE, A-6
II_GENERIC_ERROR, A-6
II_HELPDIR, A-6

II_INACTV_TMOUTINT, 7-17, 7-18, A-7
II_INSTALLATION, A-7
II_NET_LMOD, A-8
II_NO_ENQ_SUBSYS, A-8
II_PROTOCOL_CCI, A-8
II_PROTOCOL_RESTART_COUNT, A-8
II_PROTOCOL_SNA_LU0, A-8
II_PROTOCOL_SNA_LU62, A-8
II_PROTOCOL_TCP_IBM, A-8
II_PROTOCOL_TCP_IBM_BUFSIZE, A-8
II_PROTOCOL_TCP_KNET, A-9
II_PROTOCOL_TCP_SNS, A-9
II_PROTOCOL_TCP_SNS_BUFSIZE, A-9
II_PSF_POOL, A-8
II_QEP_SIZE, A-9
II_RECALL, A-9
II_SECURITY, 7-14, 7-15, 7-17, A-9
II_SMFID, A-10
II_TIMEZONE, 7-18, A-10
II_UTEXE_DEF, A-10
ING_AREA, 8-1, 8-5, 8-6
ING_MODE, A-10, A-13
ING_SORT, 8-1, 8-5, 8-6
LOCK_LISTS, A-11
LOCK_MAX, A-11
LOCK_RESTAB, A-11
LOCK_TABSIZE, A-11
LOG_CPINTVL, A-11
LOG_SBLKS, A-11
MAX_LOCKS, A-11
SABExxxx, A-13
SRV_MAXSERVERS, A-11
SRV_MAXTHREADS, A-11
SRV_STKSIZE, A-12
SRV_TMOUTINT, 7-18, A-12
SYS_INGRES, A-12

Logical symbols
 customizing, 4-29, A-2
 database locations, A-3
 descriptions, A-2
 library, A-1
 location of, 8-3
 syntax, A-2
 user, 8-1, 8-5

Logical units, 7-5

M

Mapping

 of data types, 9-6
 redefined storage, 9-27

Master database rebuild, 10-22
MAX_LOCKS logical, A-11
Microsoft FrontPage, B-1
Microsoft Internet Information Server (IIS), B-1
Microsoft Query, D-1
Modes of access, 1-5
modify (command), 10-3
Money data type, 9-21
multiple gateways
 installing, F-1

N

Name server

 allocating files, 4-30, 4-31
 files, B-3
 function of, 2-4

Net

 described, 1-5
 I/O, 10-7
 initializing, 4-33
 installing, 3-7
 parameters, 4-29
 server, 2-4, 2-5
 specifying logical virtual node, A-6
 utilities, 6-32

netu (utility)

 configuring server as client, 10-31
 establishing remote authorization, 7-13, 7-15, 7-16
 main menu, 6-32

No server mode, A-10

Non-relational

 gateways, 1-2
 parameter, A-11, A-13

notrace (command), 10-5

Null values, 4-9

Numeric data types, 9-12

O

Objects

sharing, 9-29

ODBC

data source, C-1, D-1
data sources, B-4
data sources in MS Windows, C-5
Database Results Wizard, B-3
driver, C-1
interface, B-1

OpenSQL

described, 1-8

OS/390

access to resources, 3-6
commands, 10-3
libraries, 3-5
logical symbols, A-4
requirements, 3-1
subsystem name, A-5
utilities, 3-5

P

Parameters

customizing, 4-29
specifying, 4-11

Partial-key searches, 9-23

Partitions

checking access to, 8-10
creating for user, 6-28, 8-8
creation utility, 10-19
deleting, 10-21
diagram, 2-8
gateway, 10-10
overview, 8-3
storing table templates, 2-6
verifying, 10-25
virtual storage requirements, 3-3

PL/I

converting data types, 9-8
creating JCL to compile, 6-39

Port ID

KNET TCP/IP, 4-15

Privileges, granting, 9-28

Product tape, 4-3

Protocol server

activate, 10-4
function of, 2-4
inactivate, 10-4
specifying parameters, 4-15
tracing, 10-4

Q

Query

Execution Plan, 10-35
languages, 1-8
Optimizer, 10-33

R

RACF, 7-13, B-2

Range searches, 9-22

Read access to data, 8-2

Record layouts, redefining, 9-27

Recovery, 10-8

Redefined storage, 9-27

Register

data, 2-5, 9-1
table scripts, building, 9-30

register table (statement), 9-2, 9-5, 9-6, 9-22, 9-23

Registration

granting privileges, 9-28
problems, 9-32
segments, 9-30
VANTAGE Objects, 3-8
VSAM data sets, 9-2

Relational

gateways, 1-2
tables, 9-5

Remote

mode, 1-5

remove table (command), 9-33

Repeating group access

table registration, 9-24

Resources

control, 10-28

- of DBAs, 5-1
- required, 10-39

Routing exits, 10-51

S

SABE member, A-1, A-13

Security

- facilities, 7-13
- inactive user timeout feature, 7-17
- specifying logical facilities, A-9
- supported systems, 1-7, 3-2

Security interfaces

- enabling, 7-12
- testing, 7-12

Sense code, 7-6

Server

- address space, 3-3, 10-5
- Communication, 2-4, 2-5
- configuring, 10-31
- connecting two, 10-31
- creating an additional, 10-28
- creating RACF profile, 7-13
- customizing, B-1
- described, 1-5
- maintaining address space, 10-3
- mode, A-10
- multiple gateways, 2-4
- Net, 2-4, 2-5
- specifying default class, A-6
- starting and stopping, 10-1
- starting as a batch job, 10-3
- timeout, A-12

Sharing

- areas, 8-3
- objects, 9-29

Single-user mode, A-13

Size of area data sets, 8-3

SMF record number, A-10

SNA LU0

- abend codes and messages, 7-3
- specifying logical, A-8
- specifying parameters, 4-15

SNA LU0 for OS/390
configuration, 7-1

SNA LU62

- specifying logical, A-8
- specifying parameters, 4-15

SNA LU62 for OS/390

- configuration, 7-4

SNS/TCP, 7-11

Software requirements, 3-1

Sort

- buffers, specifying logical number, A-3
- work data set, 8-3
- work file, 2-7

sort criteria

- defining, D-4

SQL

- described, 1-8
- overview, 1-8
- scripts, 10-11

SRV_MAXSERVERS logical, A-11

SRV_MAXTHREADS logical, A-11

SRV_STKSIZE logical, A-12

SRV_TMOUTINT logical, 7-18, A-12

Standard system catalogs, 2-6

Star, 1-7

Starting

- gateway server, 10-1
- server as batch job, 10-3
- session as batch program, 6-12

Statements, 1-8

Storage requirements, 3-2, 4-4

Structured Query Language, 1-8

Syntax documentation conventions, 1-9

SYS_INGRES logical, A-12

System

- catalogs, 2-6
- libraries, 3-5
- utilities, 3-6

T

Table names, relational, 9-5

tables

- selecting, D-2
- Tables
 - alternate translation, 7-19
- Template directory, specifying logical name, A-4
- Terminal monitor
 - help files, A-6
 - invoking, 6-14
 - using, 6-12
- Terminology
 - documentation conventions, 1-8
- Testing access to registered table, 9-31
- Threads
 - active, 10-5, 10-8
 - concurrent, A-11
 - gateway, 2-2, 2-4, 2-5
 - idle, 10-8
 - maximum, 10-8
- Time zones
 - local, 7-18
- Timeout
 - force inactivate, 7-17
 - of Optimizer, 10-49
 - server, A-12
- trace (command), 10-4
- Trace points, 10-51
- Troubleshooting, 10-3
- TSO
 - access to, 8-1
 - address space, 2-5, 3-3, 10-31
 - logon procedure, 4-32

U

- U member, A-2
- User
 - accessing objects, 9-29
 - adding, 8-4
 - context, 10-4, 10-6
 - creating default, 4-11
 - creating entries, 4-29
 - defining logical symbols for, 8-5
 - formatting area, 8-6
 - ID, 4-19, 8-1
 - inactivate, 10-4

- needs, 8-2
- partition, 6-28, 6-30, 8-8
- setup, 3-8, 8-1
- specifying, 4-11

- User interfaces
 - initializing logical, A-10
 - supported, 1-8
 - version required, 3-2

- utilities
 - Database Object Manager, C-3, C-5

- Utilities
 - database, 10-19
 - defined, 1-8
 - logical, A-11

V

- Vantage
 - accessing data, C-1, D-1, B-1
 - displaying data, D-5, B-1

- Varchar data type, 9-10, 9-11

- verifydb (utility), 10-25

- Verifying
 - functionality, 3-7
 - installation, 3-6, 5-1

- VTAM
 - configuration, 7-6
 - SNA APPL definition, B-3

- vwconst (utility), 10-28

W

- web page
 - creating, B-3

- Whitesmiths C, 6-36

- Write access to data, 8-2

Y

- Year 2000 support, 7-19

