Ingres 10.0

System Administrator Guide

INGRES

Contents

Chapter 3: Configuring Ingres   37

Chapter 4: Setting Environment Variables and Logicals 49

Chapter 5: Customizing Ingres 83

Chapter 6: Troubleshooting Ingres 95

Chapter 7: Using Monitoring and Tracing Tools 173

Chapter 8: Managing Your System and Monitoring Performance   213

Chapter 9: Analyzing and Recovering Journaled Transactions 227

Chapter 10: Understanding Ingres Management Architecture 243

Appendix A: Environment Variables and Logicals   341

Chapter 1: Introducing Ingres System Administration

This section contains the following topics:

Audience

The *System Administrator Guide* is designed for the system administrator who has overall responsibility for the operation of the Ingres® installation. The system administrator must have a solid working knowledge of operating system management or be in close contact with the operating system administrator.

System Administrator Responsibilities

The system administrator has all privileges and holds the primary responsibility for installing and maintaining Ingres. The system administrator (often referred to as the installation owner) owns the user ID specified during Ingres installation. Logging in as this user provides permissions that are needed for installation and maintenance.

The system administrator has the following responsibilities:

- Installing Ingres (and optional components)

- Authorizing users to access Ingres

- Defining Ingres variables such as II_DATABASE and II_INSTALLATION

- Starting, stopping, configuring, and monitoring servers

- Disconnecting or suspending a session connected to a server

- Shutting down the Ingres installation (or its components)

System Administration Summary

The following table is a quick reference to the commands or utilities used to perform system administration tasks. Refer to the appropriate chapter or guide for details. Most of these tasks can be also performed using the Visual Tools on Windows.

Task	Command or Utility
Start Ingres	ingstart
Stop Ingres	ingstop
Set Ingres configuration parameters	cbf
List, set, or unset Ingres installation variables	ingprenv
	ingsetenv
	ingunset
Monitor Ingres installation	ipm
Configure Ingres Net	netutil
Grant installation privileges (For details, see the *Security Guide*.)	GRANT...ON CURRENT INSTALLATION
Display information on process and system resources	syscheck

Special Considerations

Before using this guide, you should be aware of the following issues.

Ingres installations can be administered in compliance with the C2 security standard.

If you are using an Enterprise Access product, see your Enterprise Access documentation for information about syntax that may differ from that described in this guide.

Ingres is compliant with ISO Entry SQL92. In addition, numerous vendor extensions are included. For details about the settings required to operate in compliance with ISO Entry SQL92, see the *SQL Reference Guide*.

Query Language Used in this Guide

The industry standard query language, SQL, is used as the standard query language throughout this guide.

QUEL commands and system parameters, such as environment variables or logicals, are also included in this guide. For a complete description of available QUEL statements, see the *QUEL Reference Guide*.

System-specific Text in this Guide

Generally, Ingres operates the same way on all systems. When necessary, however, this guide provides information specific to your operating system. For example:

UNIX: Information is specific to the UNIX environment.

VMS: Information is specific to the VMS environment.

Windows: Information is specific to the Windows environment.

When necessary for clarity, the symbol ▼ is used to indicate the end of system-specific text.

For sections that pertain to one system only, the system is indicated in the section title.

Path Notation in this Guide

The directory structure of an Ingres installation is the same regardless of operating system. Rather than showing path examples for all environments, this guide uses UNIX notation only.

For example: When describing the location of the collation sequence file, the guide shows: $II_SYSTEM/ingres/files/collation/collation_name.

On Windows, the location is:
%II_SYSTEM%\ingres\files\collation\collation_name

On VMS, the location is:
II_SYSTEM:[INGRES.FILES.COLLATION]collation_name

Terminology Used in this Guide

This guide uses the following 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 execute interactively from a terminal monitor.

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

Syntax Conventions Used in this Guide

This guide uses the following conventions to describe syntax:

Convention	Usage
Monospace	Indicates key words, symbols, or punctuation that you must enter as shown
Italics	Represent a variable name for which you must supply an actual value
[ ] (brackets)	Indicate an optional item
{ } (braces)	Indicate an optional item that you can repeat as many times as appropriate
	(vertical bar)	Separates items in a list and indicates that you must choose one item

Chapter 2: Understanding Ingres Components

This section contains the following topics:

This chapter discusses the Ingres components, utilities, and tools available to the system administrator to work with Ingres.

Components of an Ingres Installation

The system administrator installs and maintains the following principal components or associated family of compatible components:

- DBMS Server

- General Communication Facility

- Querying and reporting tools

- Visual tools, if available in your environment

- Logging and locking systems

An Ingres installation also has:

- Library files and utilities, provided with the product distribution

- Configuration files and error log files, created during installation and at runtime by Ingres

- Databases and their associated files: the checkpoint, journal, dump, and work files, also created by Ingres during installation and at runtime

The locations of these files are selected at installation time, together with the configuration of the Ingres DBMS Servers and the logging and locking systems. For the initial installation, default settings are provided by the system whenever possible.

Ingres Architecture
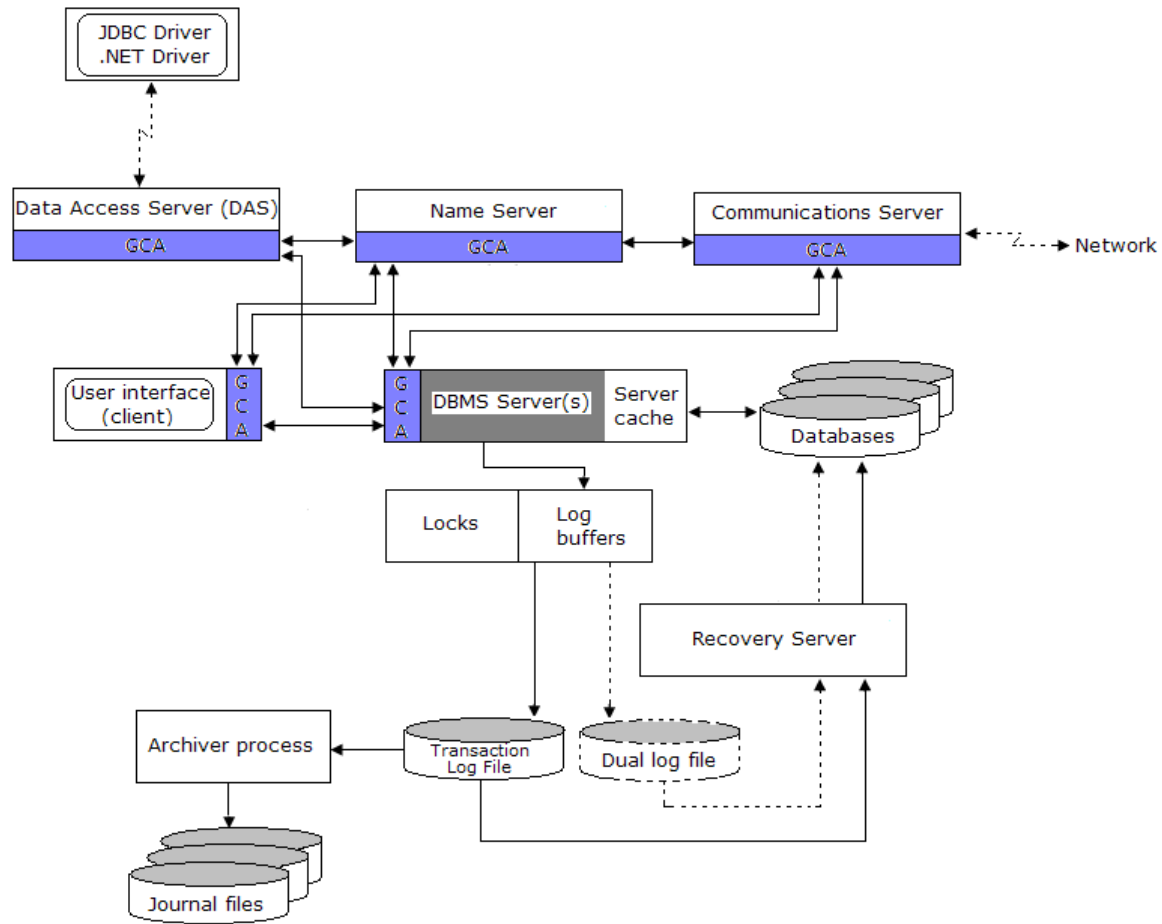
DBMS Server

The Ingres client-server architecture allows multiple users access to databases through connections to one or more DBMS Server processes. The DBMS Server (iidbms) is a multi-threaded daemon process that performs asynchronous disk input and output. The number of users connected to a server is limited by hardware or the operating system.

Some elements of the DBMS Server are described as facilities (see page 20).

Query Processing

A typical interaction with Ingres consists of at least two processes:

- An Ingres tool or application
- DBMS Server process

The Ingres Tool or Application Process

The Ingres tool or application performs the following actions:

- Takes user input and issues a query that is sent to the DBMS Server.
- Formats, optimizes, and executes the query on behalf of the user.
- Displays the resulted data returned by the DBMS Server.

DBMS Server Process

The Ingres DBMS Server is a multi-threaded process. It can execute queries for a large number of users, each running an Ingres tool. Even though it is a single process, the DBMS Server can execute queries as multiple "sessions" on behalf of multiple users. You can view which sessions are running in the DBMS Server at any moment by using the ipm, iimonitor, and Visual Performance Monitor tools.

Query Environment

When a thread or session executes a query inside the DBMS Server, it does so in a *query environment*. The query environment consists of:

- A quantity of resources available from the operating system for use by the session.
- The rules under which the query is executed. These rules reflect which query language is used, which locking strategy is employed, which diagnostic information is returned, which default behavior Ingres adopts for various query language statements, and so on.

DBMS Server Facilities

The following are DBMS Server facilities, which are configured using DBMS Server configuration parameters:

- Abstract Data Type Facility

- Compatibility Library Facility

- Data Manipulation Facility

- Optimizer Facility

- Parser Facility

- Query Execution Facility

- Query Storage Facility

- System Control Facility

- Relation Description Facility

- General Communication Facility

Abstract Data Type Facility

Abstract Data Type Facility (ADF) does all the work that involves data types. It manipulates floating-point numbers, character strings, integers, and all the conversions and comparisons between them. This facility can be executed independently from the server so that other Ingres tools can also use ADF.

This abstract data type facility is also compiled into the Ingres tools such as QBF (Query-By-Forms) and the Terminal Monitor. It is used to manipulate Ingres data types in the front-end process before sending them to the DBMS Server.

Compatibility Library Facility

Compatibility Library Facility (CLF) provides the Ingres DBMS Server with a platform independent interface to the operating system. It contains functions that perform services such as disk I/O, memory management, Semaphore manipulation, and context switching (when Ingres threading is configured). It also provides low level system communications functions.

For example, when a thread in the DBMS is idle, it normally shows up in iimonitor, IPM, or VDBA as having current facility CLF. This indicates that the thread is executing a CLF function to read information from the communications channel that connects the thread to the system process that instigated the database connection associated with this thread.

Data Manipulation Facility

The Data Manipulation Facility (DMF) manages the DBMS Server interface to disk storage. In addition to managing all storage structures (hash, heap, ISAM, B-tree, BLOBs, and so on), DMF uses the logging and locking systems to control transaction processing and to handle concurrency issues. Included within DMF is a buffer manager that controls access to a cache (possibly shared) of database pages.

Optimizer Facility

The Optimizer Facility (OPF) selects the optimal plan for implementing queries. It also converts the query tree that comes out of the parser into the query execution plan (QEP). OPF uses a memory pool for its operations.

Parser Facility

The Parser Facility (PSF) converts queries from text form to internal format. The parser adds data from the system catalogs to the query, such as information about the table structure and keys that the optimizer needs to make a useful query plan.

Query Execution Facility

The Query Execution Facility (QEF) executes query plans and database utilities. It provides internal query services that other facilities use. QEF manages repeat queries, transactions, and cursors.

Query Storage Facility

The Query Storage Facility (QSF) provides shared memory facilities with a temporary or permanent place to store query trees and query plans.

System Control Facility

The System Control Facility (SCF) is the central controlling facility that manages sessions on behalf of user (client) requests. It coordinates the actions among the various facilities involved in processing a query, including thread monitoring and switching. SCF is also responsible for managing server-wide access to shared resources, such as operating system semaphores and memory.

Relation Description Facility

The Relation Description Facility (RDF) is a central caching point for information about tables. RDF is used by Ingres Star, PSF, and OPF.

General Communication Facility

The General Communication Facility (GCF) manages communication among all components of Ingres. GCF has five elements:

- Name Server
- Communications Server
- Data Access Server
- Bridge Server
- General Communications Architecture

Name Server

The Name Server keeps track of all DBMS, Data Access, Communications, Bridge, Star, and Enterprise Access servers in an installation and provides user processes with the information they require to connect to local or remote installations.

How the Name Server Works

The Name Server provides information to user processes that enables a connection to a local DBMS Server. When a process wants to connect to a remote DBMS Server:

- The Name Server provides information that allows the process to first connect to a Communications Server.
- The Communications Server establishes communication with the remote DBMS Server.
- The Name Server checks regularly (the default is every five minutes) to ensure that all DBMS Servers on its list are functioning. If a server has shut down, the Name Server automatically deletes its registration.

Name Server Maintenance Utility

The Name Server Maintenance Utility (iinamu) is a part of the General Communication Facility provided by Ingres. This utility allows the system administrator to manually add servers to or delete servers from the list maintained by the Name Server, or to stop and restart the Name Server.

For more information about this utility, see the *Command Reference Guide*.

Communications Server

The Communications Server (iigcc) is a daemon process that provides the network communication element of the Ingres Net product. It monitors outgoing communication from local applications to remote DBMS Servers and incoming communication from remote applications to local DBMS Servers. An installation can have multiple Communications Server processes.

For more information on the Communications Server, see the *Connectivity Guide*.

Data Access Server

The Data Access Server enables communication between Java applets and applications and Ingres. The Data Access Server is the server component of the JDBC Driver, and also provides DBMS Server access for the Ingres .NET Data Provider. Data Access Server parameters include the maximum number of inbound sessions, client connections, connection pooling, pooling timeouts, network port, listen address, and protocol status. Configuration of the Data Access Server is done at installation and rarely requires reconfiguration.

Bridge Server

The Ingres Bridge extension enables communication between Ingres clients on one network and servers on a different type of network. Reconfiguration of the Bridge Server is rarely required.

For information on the Bridge Server's role in Ingres connectivity, see the *Connectivity Guide*.

General Communications Architecture

The General Communications Architecture (GCA) is the lowest level General Communications Facility Application Program Interface. The GCA maintains communication connections between processes on the same local Ingres installation. The GCA is a subroutine library that is a part of all Ingres tools, DBMS Servers, Star Servers, and the libraries associated with embedded SQL and EQUEL.

Querying and Reporting Tools

Ingres provides several querying and reporting tools such as Report-Writer, Report-By-Forms (RBF), Query-By-Forms (QBF), and Vision to enable users to access Ingres databases.

When a user accesses a database through tools like Report-Writer, Report-By-Forms, Query-By-Forms, or Vision, a process is created and communication is initiated with the database through a DBMS Server. The Name Server, which is a part of the General Communication Facility, provides user information.

For more information on Ingres querying and reporting tools, see the *Character-based Querying and Reporting Tools User Guide*.

Visual Tools

On Windows, Ingres provide various tools with a graphical user interface (GUI) that let you start, stop, manage, and monitor the Ingres installation.

Visual Manager

Ingres Visual Manager (IVM) provides a global view into the Ingres installation. It serves as a system console from which you can manage Ingres components and access other utilities. This utility captures events that are occurring in the system and allows them to be filtered for emphasis, based on your preferences.

You can access all other Ingres visual tools through IVM. For an overview of IVM tasks, see the chapter "Managing your System and Monitoring Performance." For instructions on using IVM screens, see the IVM online help. For information on IVM command line tasks, see the *Command Reference Guide*.

Visual Performance Monitor

Visual Performance Monitor can be used as a monitoring tool, a performance analysis tool, and a system management tool.

Visual Performance Monitor lets you monitor the following:

- Servers
- Users and sessions
- Logging
- Locking
- System performance
- Replication

Visual Performance Monitor lets you perform the following actions:

- Refresh data in the window from a server
- Shut down a database server or close a session

For more information, see Visual Performance Monitor (see page 221).

Visual DBA

Visual DBA provides an alternative set of system administration tools. It allows you to perform system administrator functions including configuring, performance monitoring, backup and recovery, and remote database optimization. For an overview of Visual DBA tasks related to communications, see the *Connectivity Guide*. For instructions on using Visual DBA screens, see the Visual DBA online help. For information on Visual DBA command line tasks, see the *Command Reference Guide*

Remote Command Server

Some options and dialogs in Visual DBA result in the launch of an operating system level command (such as createdb or destroydb) on the server. These "remote commands" are performed by the Remote Command Server (RMCMD) process on behalf of Visual DBA.

The Remote Command Server (RMCMD) allows remote execution of operating system commands. It must be started in installations where a DBMS server is running for certain DBA tasks to be accessible remotely with Visual DBA. The Remote Command Server is started on the server side, not on the Visual DBA client.

DBA tasks are primarily those that do not have an equivalent through an SQL statement, for example, creating or dropping a database, displaying selected portions of the journal for a database, or starting a replication server remotely. By default, only the installation owner is authorized to perform such tasks remotely through Visual DBA.

The rmcmd process is started and stopped with Ingres, according to the settings for the Remote Command component in CBF or Configuration Manager.

Grant Access to Remote Users

To allow a user other than the installation owner to execute remote commands, you can use either of the following methods:

- You use the Create or Alter User dialog in VDBA. Enable the Remote Command (rmcmd) Privileges checkbox.

- You can execute the following SQL statements, while connected to the imadb database as the installation owner:

```
grant select,insert,update,delete on $ingres.remotecmdinview to user
grant select,insert,update,delete on $ingres.remotecmdoutview to user
grant select,insert,update,delete on $ingres.remotecmdview to user
grant execute on procedure $ingres.launchremotecmd to user
grant execute on procedure $ingres.sendrmcmdinput to user
grant register, raise on dbevent $ingres.rmcmdcmdend to user
grant register, raise on dbevent $ingres.rmcmdnewcmd to user
grant register, raise on dbevent $ingres.rmcmdnewinputline to user
grant register, raise on dbevent $ingres.rmcmdnewoutputline to user
grant register, raise on dbevent $ingres.rmcmdstp to user
```

Note: Grants must not be made directly to the underlying tables.

How Remote Commands Are Executed

Operating system commands executed by the Remote Command (rmcmd) Server are executed on the server side under the user ID that launched that server (typically the installation owner).

For commands that support the –u option, the rmcmd server will accept the command execution request only if it contains a –u*username* argument, where *username* is the user ID of the session through which the client application (typically VDBA) does the request.

For commands that do not support the –u option, such as sysmod, alterdb, or relocatedb, the request will be accepted only if the user ID of the session is the same as the user who launched the rmcmd server on the server side.

Configuration Manager

The Configuration Manager utility provides a GUI interface for configuring the Ingres installation. For more information, see Configuration Manager online help. For information on Configuration Manager command line tasks, see "vcbf" in the *Command Reference Guide*.

The Configuration Manager lets you perform the same tasks as the Configuration-By-Forms utility.

Network Utility

The Network Utility is a stand-alone tool that allows you to view and define virtual node (vnode) definitions, which are used to connect to remote Ingres installations through Ingres Net. The Network Utility provides all the same functionality as the virtual nodes toolbar in Visual DBA. The Network Utility also enables launching of the Database Object Manager, Visual Performance Monitor, and Visual SQL utilities for such installations.

For more information, see the *Connectivity Guide* and the Network Utility online help.

Export Assistant

The Export Assistant allows you to export Ingres or Enterprise Access table data into external file formats such as CSV, XML or DBF. It can be run as a stand-alone utility or from within Visual DBA. For instructions on using the Export Assistant, see the Export Assistant online help.

Import Assistant

The Import Assistant allows you to import an external text file into an Ingres or Enterprise Access table. It can be run as a stand-alone utility or from within Visual DBA. For instructions on using the Import Assistant, see the Import Assistant online help.

Journal Analyzer

The *Journal Analyzer* is a utility that allows you to view and analyze journaled transactions and individual underlying statements. It also enables you to create SQL scripts, which are used to recover or redo individual row operations without redoing or rolling back the whole database or table.

For instructions on using the Journal Analyzer, see the chapter "Analyzing and Recovering Journaled Transactions" and the Journal Analyzer online help.

Visual SQL

The Visual SQL utility allows you to enter and execute SQL queries. As part of Visual SQL, the SQL Assistant helps you build SQL queries. Visual SQL can be run as a stand-alone utility or from within Visual DBA. For instructions on using this utility, see the Visual SQL online help.

Visual Configuration Differences Analyzer

The Visual Configuration Differences Analyzer (VCDA) allows you to compare configuration snapshots of an Ingres installation.

To learn about the benefits of this tool, see chapter "Managing Your System and Monitoring Performance". For instructions on using this tool, see the Visual Configuration Differences Analyzer online help.

Visual Database Objects Differences Analyzer

The Visual Database Objects Differences Analyzer (VDDA) allows you to compare the definition of Ingres database objects that are either currently installed or saved in a snapshot file. For instructions on using this tool, see the VDDA online help.

Logging and Locking Systems

The logging and locking systems coordinate the locking, recovery, and journaling of databases. The system is composed of the following components:

- Lock manager

- Logging facility

- Recovery process

- Archiver process

- Log files

Lock Manager

The locking component controls concurrent access to a database. Shared memory is allocated to components initially during the installation procedure. The amount of shared memory that your installation requires depends on the logging and DBMS Server parameters that you select during installation.

UNIX: In UNIX, the lock manager component makes use of the shared memory segments and semaphore resources that you install when you configure the UNIX kernel.

Logging Facility

The logging facility implements a circular "write-ahead" transaction log file for the management of transactions within the installation. It ensures that log records are written in a way that makes them accessible to the recovery and archiver processes.

Log Buffers

The logging facility maintains in-memory log buffers. For every in-memory log buffer, there is a corresponding disk log file. Memory log buffers and disk log file blocks are the same length. The number of log buffers is configurable, set according to the performance requirements of the system. The logging system manages multiple asynchronous writes of log buffers to the log file.

The number of disk log file blocks corresponds to the size of the log file, and is specified when the log file is created. In a properly tuned system, most log file buffers are completely full of log records when they are written to the log file. However, as all log records associated with a transaction must be forced to the log file at certain times, principally at end transaction time, a small percentage of log file blocks may contain unused space. Server group commit logic is designed to minimize the frequency of log force operations and increase log file space use.

How the Logging Facility Works

All servers in the Ingres installation, as well as the recovery and archiver processes, share the logging facility. Log records written by different servers, or written by several threads in the same server, can be combined with log records written by other servers or threads. The Logging Facility performs the following actions:

- Copies the Log records to in-memory log buffers, which are written to the log file as they fill.

- The log file block is rewritten when the file wraps around.

Recovery Process

The recovery process performs recovery in the event of a server or system failure. The recovery process maintains a history of important actions in its own message log file, iircp.log.

Data Manipulation Facility Recovery Process (dmfrcp)

Each installation of Ingres has a dmfrcp (data manipulation facility recovery process) that is assigned recovery responsibilities. In normal circumstances, transaction commit and rollback processing are handled within each DBMS server. In the event of a server or system failure, however, the recovery process performs the required recovery, including backing out uncommitted transactions and ensuring that committed transactions are properly reflected on disk.

Recovery Modes

The recovery process operates in the following modes:

- Online recovery
- Offline recovery

Online Recovery

Online recovery is performed when a server stops abnormally. In this case, users connected to other servers are generally unaffected by the recovery.

Offline Recovery

Offline recovery is performed when the installation is brought back up after it has stopped abnormally. In this case, the installation remains unavailable until the recovery process completes all required recovery.

Archiver Process

The archiver process (dmfacp) is responsible for copying the history of operations performed on journaled databases from the transaction log file to the journal files. The archiver process maintains a history of important actions in its own message log file, iiacp.log.

The archiver process removes completed transactions from the transaction log file. For journaled tables, the archiver writes any completed transaction to the journal files for the database. Each database has its own journal files, which contain a record of all the changes made to the database since the last checkpoint was taken. The archiver process "sleeps" until sufficient portions of the transaction log file are ready to be archived or until the last user exits from a database.

Data Manipulation Facility Archiver Process (dmfacp)

Each installation has a single archiver process called the dmfacp (data manipulation facility archiver process). The archiver process is responsible for copying the history of operations performed on journaled databases from the transaction log file to the journal files. Journal files contain the subset of transaction log file information associated with a specific database.

The name of the DMFACP process depends on the installation code. For Ingres installations with installation code II, the process name is DMFACP. For installation codes other than II, the DMFACP process name is DMFACPxx where xx is the installation code.

Journal Files

Journal files contain the subset of transaction log file information associated with a specific database. Journal files are created (and optionally destroyed) during a checkpoint operation. In the event of a disaster, the database can be rebuilt by restoring the latest checkpoint and applying journal file information.

Disaster Recovery Command

Disaster recovery operations are coordinated by the rollforwarddb command.

Transactions Recovery

Transaction recovery involves the transaction log file that is used as a write-ahead log, plus journal files maintained on a per-database basis. Log files contain short-term recovery information regarding active databases, while the journal files contain long-term information used for auditing and disaster recovery. While the log file is circular and wraps around, journal files are of configurable length and are retained indefinitely.

Ingres employs a page-oriented recovery scheme, where changes to pages are reflected in the transaction log file.

Types of Transaction Recovery

Recovery information is divided into two types:

- Undo (or Rollback) Operations
- Redo (or Cache Restore) Operations

Ingres performs both online and offline recovery, as described in Recovery Modes (see page 30).

Undo Operation

Undo or transaction backout recovery is performed by the DBMS Server. For example, when a transaction is aborted, transaction log file information is used to roll back all related updates. The DBMS Server writes the Compensation Log Records (CLRs) to record a history of the actions taken during undo operations.

Redo Operation

A Redo recovery operation is database-oriented. Redo recovery is performed after a server or an installation fails. Its main purpose is to recover the contents of the DMF cached data pages that are lost when a fast-commit server fails. Redo recovery is performed by the recovery process. Redo recovery precedes undo recovery.

Redo Operation in a Cluster Environment

In an Ingres cluster environment where all nodes are active, the local recovery server performs transaction redo/undo for a failed DBMS server on its node, just like in the non-cluster case. The difference in a cluster installation is that if the recovery process (RCP) dies on one node, either because of an Ingres failure, or a general failure of the hardware, an RCP on another node will take responsibility for cleaning up transactions for the failed nodes.

Ingres Log Files

Ingres maintains several log files to which it writes information about the installation activities. The various log files are:

- Transaction Log File
- Error Log
- Archiver Log
- Recovery Log
- VMS Cluster Service Process Log
- Primary Configuration Log Files
- Operational Configuration Log Files
- Optional Log Files

Transaction Log File

Each installation has a transaction log file, and an optional dual log file. The transaction log file holds information about all open transactions and is used to recover active databases after a system failure. You have the option to change its size and number of partitions at startup.

UNIX: The UNIX log file can be created as a raw partition or as a number of raw partitions. For details, see the *Installation Guide*.

Error Log

The Error Log (errlog.log) file is the main log file. It is a readable text file that you can use for troubleshooting. Messages about the installation are appended to this log with the date and time at which the error occurred. This is generally the first place to look when troubleshooting a problem. The error log contains the following information:

- Error messages
- Warning messages
- Server start up and shutdown messages

The system administrator maintains the error log file. This file continues to grow until manually truncated. The installation must be shut down before truncating or removing the errlog.log file.

The error log file is located in the $II_SYSTEM/ingres/files/ directory.

Archiver Log

The archiver log (iiacp.log) contains information about the current archiver process. The archiver log file is appended to when the archiver process starts. The log contains the following information:

- Archiver start up
- Archiver error messages
- Archiver warning messages

The archiver log is located in the $II_SYSTEM/ingres/files/ directory.

Recovery Log

The recovery log (iircp.log) contains information about the current recovery process. This file is appended to when the recovery process starts. The log contains the following information:

- Current logging and locking parameter values
- Recovery process error messages
- Recovery process warning messages
- Recovery operations information

Note: The recovery log must be monitored if you are unable to connect to Ingres and suspect that the DBMS Server is in recovery mode.

The recovery log is located in the $II_SYSTEM/ingres/files/ directory.

Primary Configuration Log Files

Ingres maintains transcripts of various configuration operations. The primary configuration log files that Ingres uses are as follows:

config.log

The config.log file contains a log of the changes made with the Configuration Manager or Configuration-By-Forms (cbf) utility.

rcpconfig.log

The rcpconfig.log file contains log and error information of the last time rcpconfig was run. For details of the rcpconfig command, see the *Command Reference Guide*.

The primary configuration log files are located in the $II_SYSTEM/ingres/files/ directory.

Optional Configuration Log Files

Ingres provides the following optional configuration log files. These files, if present, are in the directory indicated by II_CONFIG.

Note: In UNIX, the II_CONFIG file must always be in $II_SYSTEM/ingres/files.

iilink.log

Contains a log of the last time the iilink command was run. This log is used only in conjunction with the Object Management Extension.

Note: This log exists on Windows only.

iivdb.log

Contains a transcript of the last time verifydb was used to diagnose or attempt recovery of a damaged or inconsistent database. This file is created the first time verifydb is run.

Note: This log does not exist on Linux.

lartool.log

Contains a transcript, with any errors, of the last time lartool was used to manually abort or commit a running transaction. This file is created the first time lartool is run.

Optional Log Files

Ingres provides various optional log files. These log files can be used for troubleshooting purposes. Ingres provides optional log files for:

- Logging messages for specific processes
- Generating logs for a specific Ingres facility
- Tracing logs and tracking messages at a greater level of detail

Process Logs

Process logs can be set up for the following processes to isolate the error messages relating to that process:

- DBMS
- GCC

Optional Log Facility

Logging can be specified for a specific Ingres facility. The following optional log and trace log files can be established by setting the associated Ingres variables.

DBMS Error Log File

The DBMS Server error log is optionally defined as a separate file. This log file is established by setting the Ingres variable II_DBMS_LOG to a user-defined file name.

Note: By default, all logged messages are sent to the errlog.log file. If you define separate error logs, all messages are sent to both that error log file and errlog.log.

GCC Trace Log File

The GCC trace log is set up for specific troubleshooting efforts. You set II_GCA_LOG to a user file name. The associated Ingres environment variable II_GCC_TRACE defines the level of tracing.

Star Error Log

The Star error log is optionally defined as a separate file. This log file is established by setting the Ingres variable II_STAR_LOG to a user file name.

Note: All Ingres Star errors and messages are sent to errlog.log by default.

For more information about these log files, see the chapter "Setting Environment Variables and Logicals" and the appendix "Environment Variables and Logicals."

Chapter 3: Configuring Ingres

This section contains the following topics:

This chapter shows how to get started using the Ingres configuration tools and describes configuration considerations for each Ingres component.

Configurable Components

After the installation is running, all Ingres servers must be monitored to ensure that they are running and that they are configured for top performance. You can also configure selected Ingres parameters.

You can view and set values for the following components:

- Server components

    - DBMS

    - JDBC

    - Data Access

    - Communications

    - Bridge

    - Star

    - Name

    - Recovery

- System components

  - Security

  - Locking system

  - Logging system

  - Primary and dual transaction logs

Configuration Tools

Ingres provides the following tools to configure an Ingres installation.

- Configuration-By-Forms (CBF) or the visual tool equivalent, Configuration Manager (VCBF)

- Visual Configuration Difference Analyzer (VCDA)

Access Configuration-By-Forms Utility

Configuration-By-Forms (CBF) is an interactive forms-based utility that is available in all environments in which Ingres runs. All configuration parameters are described in CBF's online help facility.

To access the Configuration-By-Forms utility

Enter **cbf** at the command line.

How You Navigate in CBF

Here are tips on how to navigate within CBF to locate and set parameters:

- On the main screen, all Ingres system components are listed. Use the arrow keys to highlight the component you want to configure.

- The menu at the bottom of each screen lists the functions you can perform. To perform a function, use its key combination shown. For example, to Configure, press SH+F1.

- To scroll through all available functions shown in the menu at the bottom of the screen, press the ESC key.

- To cancel an action, press the ESC key.

- To return to the previous screen, press F10.

- To exit CBF, press F6 from the Main Screen.

- For help on any screen or parameter, press F1.

Access Configuration Manager Utility

Configuration Manager is a visual tool that provides the equivalent functionality as Configuration-By-Forms. Configuration Manager is available on Windows only.

To access Configuration Manager

Windows: Do **one** of the following:

- Click Start on the Windows taskbar and then choose Programs, Ingres, Configuration Manager.
- In Ingres Visual Manager (IVM), click the Configuration toolbar button, or choose File, Configure.
- Right-click the database button on the system tray and choose Configuration Manager from the popup menu.
- Enter **vcbf** on the command line.

Access Visual Configuration Difference Analyzer Utility

The Visual Configuration Difference Analyzer (VCDA) allows you to compare configuration snapshots of an Ingres installation. VCDA is a visual tool available on Windows only.

To access VCDA

Windows: Do **one** of the following:

- Click the VCDA toolbar button in Ingres Visual Manager (IVM).
- Right-click the database button on the system tray and choose VCDA from the popup menu.
- Enter **vcda** on the command line.

Component Configurations

You can add, delete, or rename the various configurable server components listed in Configurable Components (see page 37).

Different configurations are needed for servers that perform different functions. For example, batch jobs can require a completely different DBMS server configuration than that of an online server. If so, create a DBMS server configuration called "batch" and configure it for batch processing.

Note: You can perform these procedures for non-default components configurations only.

Note: You cannot delete configurations for the Name Server and Recovery Server.

Configurable Properties for a Component

Each non-default component configuration has two properties that can be changed:

■ Startup Count—The startup count for a component represents the number of instances to be initiated when the Ingres installation is started.

For example, if you set the startup count to five for the Communications Server, five Communications Servers start at startup time. All components that have a startup count that can be edited are daemon processes (the DBMS Server, Star, GCC, and so on).

■ Name—Every non-default component can be assigned a name. The name allows you to assign a specific name to a component configuration.

Note: You can only rename non-default server components.

Configuration Snapshot

A configuration snapshot file contains information on the installation's parameters including:

■ Configuration parameters

■ Vnode definitions

■ System variables

■ User variables

The snapshot file can be used to compare configuration changes.

For example, if you encounter problems with the installation later, take another snapshot of the configuration and compare it to the earlier snapshot to determine if any configuration changes have contributed to the problem. Also, keep an on-going record of configuration changes by taking a snapshot of the installation each time you change its configuration.

Note: Before configuring or editing the system parameters, you must use the VCDA tool to create a "snapshot" file of the current configuration.

Configuration Parameters

Ingres configuration parameters are stored in the config.dat file.

Note: Do not edit the config.dat file directly. Use Configuration-By-Forms (or its visual equivalent Configuration Manager, if available) to set parameter values.

Some Ingres parameters are dependent on, or derived from other Ingres parameters.

For instructions on configuring specific parameters, see the online help for Configuration-By-Forms or Configuration Manager.

Setting Parameter and Derived Parameter Values

For most component configurations, you can edit the value of a parameter, restore the parameter to a value that has been saved in a configuration "snapshot" file, or restore the parameter to its original installation value.

For instructions on editing and restoring parameter values, see online help.

Using VCDA to Restore Configuration Parameter

VCDA can be used to restore configuration parameter values saved in a snapshot file. For detailed steps on using this tool, see the VCDA online help.

Protecting or Recalculating Derived Parameters

The settings of some configuration parameters are derived from those of other parameters. When a parameter is changed, any parameters derived from that parameter are automatically updated.

Any derived parameter can be set in "protected" mode, which means that the Configuration-By-Forms utility or Configuration Manager cannot recalculate the derived value. Protected mode allows you to set a desired value for the parameter, even though the value is normally derived by the system from other parameters.

If protected mode is disabled, the value of the parameter is adjusted whenever new values are set for other parameters on which the derived value is based. The new and old values are shown in the change log.

For detailed steps on performing these procedures, see the online help.

Components That Use Derived Parameters

The following components contain parameters that are derived from other parameters:

- DBMS Server
- Star Server
- Security
- Locking System
- Logging System
- Recovery Server

DBMS Server Configuration

The Ingres client-server architecture allows multiple users access to databases through connections to one or more DBMS Server processes. The DBMS Server (iidbms) is a multi-threaded daemon process that performs asynchronous disk input and output. The number of users connected to a server is limited by hardware or the operating system.

Since it is unlikely that the default DBMS server configuration suits all users' requirements, the DBMS Server is usually configured to improve performance in a particular configuration. For a selected DBMS server definition, its buffer caches can be configured as well as other parameters and derived parameters.

DBMS servers can be configured to fit specific needs. For example, you can:

- Designate one server to access a particular database, denying other servers access to that database.
- Configure a server as a "fast commit" server to achieve higher levels of performance.
- Specify that two or more servers share a common memory buffer cache.
- Specify private buffer cache with distributed multi-cache protocol.

Note: Ingres preallocates most of the resources it requires. It may use hundreds of gigabytes of virtual memory and allocate millions of handles; the use of these resources depends on the number and size of the page caches enabled.

Fast Commit Option

The fast_commit option, which is a derived DBMS Server parameter, allows the server to execute commits without forcing data pages to the database. This reduces I/O in the server and improves response time.

Without fast commit, all update transactions must do the following when a commit is executed:

- Force all "log" records for this transaction to disk

- Force all updated data pages to disk

- Write a "commit" record to the "log" file and force it to disk

This ensures that the data is committed and the data in the database has been written to disk when a system crash occurs immediately following a commit.

With fast commit, a server can perform a commit by writing a commit record to the log file and forcing it to disk. The actual data pages are not written to the database disk until an optimal time to do so is reached. If necessary, the committed transactions can be recovered from the log file that was written to disk.

Fast commit can greatly improve transaction response time and throughput and greatly reduce direct I/O to the database. All update transactions gain some benefit from using fast commit because control is returned to the application while database writes are processed asynchronously. Applications that tend to update the same group of records many times gain a big increase in throughput when using fast commit because writes to the database are delayed until many transactions have actually been committed. This allows the system to write out many updates with few I/O operations.

However, using fast commit can also increase recovery times. If the system crashes, all committed transactions that have not yet had their data written to the disk must be recovered. To control this recovery time, adjust the consistency point frequency.

Write Behind Option

The dmf_write_behind option, which is a DBMS Cache parameter, allocates server threads that asynchronously write modified data pages to disk. The write behind threads are controlled by the dmf_wb_start and dmf_wb_end thresholds (both derived DBMS Cache parameters) that were defined at server startup time.

The write behind threads (dmf_write_behind) can be either ON or OFF for each defined cache (2k, 4k, and so on). The default is ON. A primary write behind thread is on for the life of the server. Additional threads are spawned to assist with the flushing of the cache, if necessary. When the flush is complete, these temporary threads are terminated.

The dmf_wb_start and dmf_wb_end thresholds (derived DBMS Cache parameters) can be individually overridden for each cache, but doing so is typically not necessary.

By allowing transactions to commit without waiting for data to be flushed to disk, fast commit can significantly improve response time. However, the data page caches of fast commit servers tend to become filled with modified pages. The write behind threads control the number of modified pages by asynchronously writing them to disk.

If write behind threads are not used, a server quickly reaches its modified page limit (specified by dmf_modify_limit). When that happens, transactions begin to do synchronous writes to free up space in the cache, negating some of the benefits of using fast commit. For this reason, if you configure a server with fast commit, dmf_write_behind must be ON to get the full performance benefits of fast commit.

For two sessions to have concurrent access to a database using fast commit, each session must access the database through the same DBMS Server or through servers connected to a common buffer. To be connected to a common buffer, the server must use the cache_sharing parameter. When a server configured with the fast_commit option opens a database, it locks the database so that no other server (except a server using the same buffer cache) can access the database.

Note: You can use the dmf_write_behind option even if you are not using the fast_commit option; write behind threads can write to the database disk even if fast commit is not specified.

Data Access Server Configuration

Data Access Server parameters include the maximum number of inbound sessions, client connections, connection pooling, pooling timeouts, network port, listen address, and protocol status. Configuration of the Data Access Server is done at installation and rarely requires reconfiguration.

For additional information on the Data Access Server's role in Ingres connectivity, see the *Connectivity Guide*.

Communications Server Configuration

Communications Server parameters include the encryption mode, logging level, registry type, number of inbound and outbound sessions, and the logging and error levels. Configuration of the Communications Server is done at installation and rarely requires reconfiguration. However, changes to a protocol or the listen address are needed. For information on the Communication Server's role in Ingres connectivity, see the *Connectivity Guide*.

Bridge Server Configuration

The Bridge Server parameters include the number of inbound sessions, the enabling of specific protocols, and a listen address.

Star Server Configuration

Star Server parameters include the number of connections supported by Star servers, accounting data maintained, and the maximum number of tables that can be kept in the cache at one time.

Distributed recovery can be disabled, meaning that the Star servers created with this definition do not recover distributed transactions at startup time. This feature is helpful if there is a problem with the installation, but you want it to continue working while the problem is being investigated. Like the DBMS Server, configuration is usually required to provide optimal performance in a given scenario.

Name Server Configuration

The Name Server parameters include the time interval for checking for active servers, and session limits. The Name Server rarely needs to be reconfigured.

Recovery Server Configuration

The configurable Recovery Server system parameters include connection limits, consistency points, events, gather write algorithms, server name, thread count, Name Server registration and stack memory allocation.

Security Configuration

You can set a variety of parameters for the Security component, including options, system parameters, security mechanisms, derived parameters, audit log file specifications, and auditing derived parameters.

Configuration of this component is required to enable security auditing and to change specifications for the audit log, which is set up during installation.

By default, an Ingres instance is set up to comply with the C2 security standard of the National Computer Security Council (NCSC). Level C2 security requires individual log-on with password and an audit mechanism.

Ingres authenticates users through the operating system account and password together with a corresponding user object definition in Ingres. Additional passwords can also be set on users and roles. For more information, see the *Database Administrator Guide*.

By default, Ingres uses the Ingres security mechanism (security_mechanism parameter). The Kerberos authentication mechanism can be used as an alternative to the Ingres security mechanism. Also, multiple mechanisms can be enabled concurrently. For details on using Kerberos, see the *Security Guide*.

Locking System Configuration

Reconfiguration of the locking system is often required. You can set the default locking level (system_lock_level) and the maximum number of locks that can be obtained per transaction (system_maxlocks). These parameters are located under the DBMS Server component. The locking parameters are installation-wide.

On UNIX systems, shared memory and semaphores are used as resources during lock control. The shared memory and semaphores used by your installation are configured in the operating system when the UNIX kernel is configured.

For detailed information on the locking system, see the *Database Administrator Guide*.

Logging System Configuration

Configurable parameters for the transaction log files include: transfer block size, buffer count, full limit, logging memory status, location, name, and number of partitions.

For a detailed explanation of the logging system, see the *Database Administrator Guide*. For instructions on creating and configuring a raw log file for UNIX platforms, see the *Installation Guide*.

Note: Before making changes to the transaction log, Ingres must be shut down using Ingres Visual Manager, Service Manager, or the ingstop command. This ensures all processes are stopped correctly and completely.

Multiple Log Partition Configuration

Ingres allows both the primary and dual transaction log files to exist on multiple devices (up to 16). Ingres improves logging system performance by allowing it to write to multiple disk devices and, on 32-bit file systems, allowing the total log file size to be greater than 2 GB.

You can configure multiple log partitions using the log_file_parts parameter. When configuring multiple log partitions, keep the following in mind:

- All log partitions must be the same size

- The primary and dual transaction logs must use the same number of partitions

For instructions on configuring multiple log partitions, see the following topic in Configuration Manager online help:

- Parameters Page, Logging System Component

Primary Transaction Log and Dual Transaction Log

If the Ingres installation is running, configurable information on the primary and dual transaction log files is read-only. If Ingres is not running, and no recovery is required from a system failure, various operations are allowed, including editing the log file name, setting the log size per file, and adding, modifying or deleting locations for the primary and dual transaction log files.

For instructions on configuring transaction log parameters, see the following topic in Configuration Manager online help:

- Primary Log Page, Transaction Log Component

Non-Configurable Components

The following components cannot be configured.

- Archiver Processes
- Remote Command Server

History of Configuration Changes

If the Ingres visual tools are available for your system, you can use the History of Changes page in Configuration Manager to view a list of all the changes to parameters that have been made by Configuration Manager or the Configuration-By-Forms (cbf) utility. The list contains changes beginning with the least recent. Each "CHANGE" line is followed by a list of all modified parameters that were derived because of the user change. Pressing F3 or Ctrl+F in Configuration Manager displays a Find dialog for searching for text.

Chapter 4: Setting Environment Variables and Logicals

This section contains the following topics:

This chapter describes the most commonly used Ingres environment variables and logicals and the system level at which they are typically set.

Scope of the Environment Variables

Ingres has a set of available logical names through which you can define various file names and other values. For an alphabetical list of the basic Ingres environment variables and logicals, see the appendix "Environment Variables and Logicals."

Ingres uses the following types of operating-system dependent environment variables and logicals:

Windows:

- Ingres environment variables

-

UNIX:

- UNIX environment variables

- Installation-wide environment variables

Ingres environment variables in UNIX and Windows are active only when you are using Ingres. A small subset of these environment variables can be reset by the user in the local environment.

VMS:

- Group-level Ingres logicals

- User-defined or process-level logicals

Ingres logicals are defined in the appropriate logical name table. ◥

Symbol Table

Most Ingres environment variables in Windows and UNIX are set in the Ingres symbol table (symbol.tbl) and are visible only with the user command ingprenv.

When setting, unsetting, or changing environment variables, a backup of the symbol.tbl file, called symbol.bak, is maintained. If the original symbol.tbl file becomes corrupted, you can use the backup symbol table file to restore it.

A history of updates to the symbol.tbl file is maintained in the symbol.log file. The symbol.log file contains information on the types of updates performed as well as some application information.

The symbol.bak and symbol.log files are located in the II_CONFIG location or the II_ADMIN location (in the case of NFS Ingres instances).

Windows Environment Variables

Windows environment variables are set at the operating system level. These take effect before, during, and after Ingres is invoked. II_SYSTEM and PATH are the only environment variables used directly by Ingres. These are set in the user's environment using the following commands entered at the operating system prompt:

```
set environment_variable=value
```

The Windows environment variables are set using Ingres Visual Manager (IVM). For more information, see the IVM online help topic, Parameters Page, Ingres Installation branch.

Alternatively, you can set Windows environment variables through the Control Panel.

UNIX Environment Variables

UNIX environment variables are set at the UNIX operating system level. These are in effect before, during, and after Ingres is invoked. II_SYSTEM, PATH, and TERM are the only UNIX environment variables required and used directly by Ingres. These environment variables are set in each user's environment using the following UNIX commands entered at the operating system prompt:

C Shell:

```
setenv environment_variable value
```

Bourne Shell:

```
environment_variable= value; export environment_variable
```

The environmental variables are typically set in a user's .login or .profile file, but can also be set in operating system startup scripts such as /etc/login.

VMS Logicals

VMS logicals are set in the system or group logical table and are visible with the system command, show logical/system or show logical/group. These are in effect before, during, and after Ingres is invoked. II_SYSTEM is the only VMS logical required and used directly by Ingres. This logical is defined by the system administrator using the following VMS command entered at the operating system prompt:

```
define [/table_level] logical_name "value"
```

The value assigned to the Ingres logical must be enclosed in double quotes.

This definition is typically placed in the operation-system wide command procedure for defining site-specific logical names (sys$manager:sylogicals.com).

Installation-Wide Environment Variables and Logicals

Ingres environment variables and logicals that are defined at the Ingres system or installation-wide level affect all users in an installation. These are usually defined during the install procedure. However, the system administrator can reset some manually.

View Environment Variables in Windows and UNIX

Use Ingres Visual Manager to view Ingres environment variables and to manually register and deregister them. Additionally, you can use it to override certain environment variables at the user level.

To view environment variables in Ingres Visual Manager:

1. Select the Ingres Installation branch in Ingres Visual Manager.

2. Select the Parameters tab and select the System subtab (if not already selected).

    The Ingres parameters that are set at the installation (system) level are displayed.

3. Select the Show Unset Parameters check box.

    Parameters that are not currently set are displayed.

    If the User subtab is selected, those parameters that can be overridden at the user level are displayed.

4. Select the Extra subtab.

    Any extra Ingres environment variables that are added or variables whose names are dynamic are displayed. You can edit these parameters from this screen.

Alternatively, to view all installation-wide environment variables from the command line, enter the following command at the operating system prompt:

```
ingprenv
```

To manually register or to remove an environment variable from the Ingres symbol table, use the ingsetenv and ingunset utilities. Always use these commands to alter the symbol table. Never edit this file directly.

A similar set of system administrator commands are available to set, print, and unset installation-wide Ingres environment variables simultaneously for all clients of an Ingres installation server. These commands are ingsetall, ingprall, and ingunsetall.

View Environment Variables in VMS

Ingres logicals (with the exception of II_SYSTEM) are defined in II_CONFIG:config.dat during the install process. These logicals can be redefined locally using the operating system command define/job logical_name "value." However, to update the logical definitions permanently, the II_CONFIG:config.dat file must be manually edited and the logical definitions modified.

To facilitate user access to Ingres tools and logical definitions, add the following access commands to the users' login.com file when you log into the system.

For users, add:

```
@II_SYSTEM:[ingres]ingusrdef.com
```

For database administrators, add:

```
@II_SYSTEM:[ingres]ingdbadef.com
```

For the system administrator, add:

```
@II_SYSTEM:[ingres]ingsysdef.com
```

The commands in these files provide tools access to all users in a system-level installation and to all users with the appropriate group user identification code (UIC) in a group-level installation. The Ingres logicals contained in II_CONFIG:config.dat are redefined at a job level for each user when one of the above commands is executed.

You can display all installation-wide Ingres logicals by typing the following command at the operating system prompt:

```
show logical [/system|/group|/job] ii*
```

Example: Set the ING_EDIT Variable

ING_EDIT specifies the default editor invoked by various editor commands.

Windows and UNIX:

To set ING_EDIT for an installation using Ingres Visual Manager, follow these steps:

1. Log in as the system administrator

2. In the Ingres Visual Manager, select the Ingres Installation branch.

3. Select the Parameters tab, and select the System subtab.

   The Ingres parameters that are set at the installation (system) level are displayed.

   Note: For variables that have a dynamic name, select the Extra tab instead of the System tab.

4. Select the Show Unset Parameters check box to display parameters that are not currently set.

5. Choose ING_EDIT from the Parameters list, and double-click in the value area.

6. Edit the value and press Enter.

7. Alternatively, click Add and scroll to choose the ING_EDIT variable. Enter the required value, and click OK.

To set ING_EDIT using a system command, do the following:

1. Log in as the system administrator

2. At the operating system prompt type the following:

   Windows:

   ```
   ingsetenv ING_EDIT D:\TOOLS\VI.EXE
   ```

   UNIX:

   ```
   ingsetenv ING_EDIT /usr/bin/vi
   ```

   VMS:

   ```
   define [/table_level] logical_name "value"
   ```

Note: On VMS, the value assigned to the Ingres logical must be in double quotes.

User-Definable Environment Variables and Logicals

Some Ingres environment variables and logicals can be set or reset by individual users in their local environment using operating system commands. Those set in a user's local environment supersede the Ingres environment variables and logicals set system-wide.

UNIX: A good place to set user-defined environment variables is the user's .login (for the C shell) or profile (for the Bourne Shell) file.

VMS: Individual users can define Ingres environment variables at the job or process level with DCL commands, or they can be defined in the user's login.com file.

Example: Set an Environment Variable in the Local Environment

An environment variable typically set in the user's local environment is TERM_INGRES. It specifies the termcap definition to be used by the forms system. It can be redefined locally by entering commands at the operating system prompt as in the following example:

Windows:

```
SET TERM_INGRES=IBMPC
```

UNIX:

C Shell:

```
setenv TERM_INGRES vt100f
```

Bourne Shell:

```
TERM_INGRES=vt100f export TERM_INGRES
```

VMS:

```
DEFINE TERM_INGRES "VT100F"
```

Display Current Value for Variables

A user can display the values set in their environment with the following command entered at the operating system prompt:

Windows:

SET

UNIX:

BSD:

printenv

System V:

env

Environment Variables and Logicals that Can Be Reset Locally

In general, only the following Ingres environment variables and logicals must be set in your local environment:

Windows:

- II_SYSTEM
- PATH

UNIX:

- II_SYSTEM
- PATH

VMS:

II_SYSTEM

The following Ingres environment variables and logicals can be reset by users in their local operating system shell:

- DBNAME_ING
- DBNAME_SQL_INIT
- II_4GL_DECIMAL
- II_ABF_RUNOPT
- II_AFD_TIMEOUT
- II_APPLICATION_LANGUAGE
- II_DATE_CENTURY_BOUNDARY
- II_DATE_FORMAT
- II_DBMS_SERVER
- II_DECIMAL
- II_DML_DEF
- II_EMBED_SET
- II_FRS_KEYFIND
- II_GC_REMOTE
- II_GCA_LOG
- II_GCx_TRACE
- II_HELP_EDIT
- II_LANGUAGE
- II_MONEY_FORMAT
- II_MONEY_PREC
- II_NULL_STRING
- II_PATTERN_MATCH
- II_PF_NODE
- II_POST_4GLGEN
- II_PRINTSCREEN_FILE
- II_SQL_INIT
- II_SYSTEM
- II_TEMPORARY
- II_TERMCAP_FILE
- II_TFDIR
- II_TIMEZONE_NAME

- II_TM_ON_ERROR
- II_VNODE_PATH
- II_WORK
- IIDLDIR
- ING_ABFDIR
- ING_ABFOPT1
- ING_EDIT
- ING_PRINT
- ING_SET
- ING_SET_DBNAME
- ING_SHELL
- INGRES_KEYS
- INIT_INGRES
- TERM
- TERM_INGRES

Important! *II_TIMEZONE_NAME can be reset for client installations only. Server installations must not reset this logical because it may affect date conversions from the local system time to the internal GMT-based value.*

Environment Variables and Logicals that Cannot Be Reset

The following Ingres environment variables and logicals must not be reset by users:

- II_BIND_SVC_*xx*
- II_C_COMPILER
- II_CHARSET*xx*
- II_CHECKPOINT
- II_CLIENT
- II_CONFIG
- II_CONNECT_RETRIES
- II_DATABASE
- II_DBMS_LOG
- II_DIRECT_IO
- II_DUMP
- II_ERSEND
- II_GCN*xx*_PORT
- II_INSTALLATION
- II_JOURNAL
- II_LOG_DEVICE
- II_MSGDIR
- II_NUM_SLAVES
- II_TUXEDO_LOC
- II_TUX_SHARED
- II_TUX_AS_MAX
- II_TUX_XN_MAX
- II_XA_TRACE_FILE
- ING_SYSTEM_SET

UNIX: These environment variables and logicals must not be visible using the env or printenv command.

Customization of Query Runtime Environment

Most aspects of the environment in which queries execute can be customized at runtime for a particular session. This can be done by executing one or more SET statements to customize the environment before query execution begins.

Here are some examples:

- Reduce lock conflicts when running read-only queries, as for reports:

    ```
    set lockmode session where readlock=nolock
    ```

- Display the text of the query as it is executed:

    ```
    set printqry
    ```

Ingres makes customization flexible by permitting most components of the query environment to be set by means of set statements at the startup of individual queries (sessions). This can be done by using various Ingres environment variables and logicals and startup files that permit users to control both the target and duration of set statement customization.

A full list of set statements can be found in your query language reference guide.

Syntax Rules for Startup Files and Environment Variables

Keep the following general syntax rules in mind while setting up startup files and environment variables:

- Set statements in startup files cannot contain more than one SET statement per line. Each statement must be on a line by itself.

- Set statements that are read from a file, but executed only by the single-line terminal monitors, must be terminated with "\g".

- Set statements defined directly inside Ingres environment variables and logicals (not written in a file) never contain the string "\g".

- Set statements defined directly inside Ingres environment variables and logicals (not written in a file) can contain multiple set statements separated by a semicolon up to a total length of 64 characters.

- The terminal monitor dayfile cannot contain set statements. It contains informational startup messages only.

- Ingres environment variables and logicals set with the following commands are global for the Ingres installation and affect the target of the environment variable/logical for all users in an installation:

  Windows and UNIX:

  ```
  ingsetenv
  ```

  VMS:

  For a system level installation:

  ```
  define/system
  ```

  For a group level installation:

  ```
  define/group
  ```

- Ingres environment variables and logicals set from the local user environment apply only to the user setting them. Other users are not affected. This is the case, for example, when Ingres environment variables and logicals are set interactively, or:

  Windows: Through the System icon in the Control Panel.

  UNIX: From a user's .login, .profile, or .cshrc file.

  VMS: Defined at the process level by the individual user with the VMS command define/process.

Dayfile

This name affects all users of the SQL and QUEL single-line terminal monitors.

Dayfile contains text messages only (not SET statements), which appear as an informational header whenever the terminal monitors are started up. It is located in $II_SYSTEM/ingres/files directory. The message header can be suppressed with the -d command line flag.

DBNAME_ING

This name applies to the QUEL single-line terminal monitor only. *DBNAME*_ING points to SET statements within a file that are executed whenever this Ingres environment variable takes effect.

*DBNAME*_ING affects users who connect to the database specified by *DBNAME* through the QUEL terminal monitor. If set globally with ingsetenv, it affects all users of *DBNAME*. If set interactively or locally in this user's .login, .profile, or .cshrc file it affects only this user.

Setting the *DBNAME*_ING environment variable is equivalent to users executing **\i** *filename* in the QUEL terminal monitor each time they connect to *DBNAME*.

Example: Set DBNAME_ING

To define this Ingres environment variable, use the following command syntax at the operating system prompt:

Windows:

```
set DBNAME_ING=path_to_file
```

where:

DBNAME is the name of the database and must be in uppercase.

For example:

```
set MYDB_ING= D:\usr\george\mystartfile
```

UNIX:

C Shell:

```
setenv DBNAME_ING path_to_file
```

Bourne Shell:

```
DBNAME_ING=path_to_file export DBNAME_ING
```

where:

DBNAME is the name of the database and must be in uppercase.

For example:

setenv MYDB_ING /usr/george/mystartfile

The file contains lines in the following format:

```
set lockmode session where readlock = nolock \g
```

This file contains the SET statements. If you have several set statements, separate the statements with a space. You must end the entire file with "\g".

DBNAME_SQL_ INIT

This name applies to the SQL single-line terminal monitor only. *DBNAME*_SQL_INIT points to SET statements within a file that are executed whenever this Ingres environment variable/logical takes effect.

*DBNAME*_SQL_INIT affects users who connect to the database specified by *DBNAME* through the SQL Terminal Monitor. If set globally, it affects all users of *DBNAME*. If set interactively or locally, it affects only this user.

Windows: Global setting is with ingsetenv, or locally in the user's environment.

UNIX: Global setting is with ingsetenv, or locally in the user's .login, .profile, or .cshrc file.

VMS: Global setting is with define/system, define/group, or locally with define/process in the user's login.com file.

Setting the DBNAME_SQL_INIT environment variable/logical is equivalent to a user's executing **\i** *filename* in the SQL Terminal Monitor each time they connect to *DBNAME*.

Example: Set DBNAME_SQL_INIT

To define this Ingres environment variable, use the following command syntax at the operating system prompt:

Windows:

SET *DBNAME*_SQL_INIT = *path_to_file*

where:

DBNAME is the name of the database and must be in uppercase.

For example:

SET *MYDB*_SQL_INT=*c:\user\mystartfile*

UNIX:

C Shell:

setenv DBNAME_SQL_INIT *path_to_file*

For example:

setenv MYDB_SQL_INIT /usr/george/mystartfile

Bourne Shell:

*DBNAME*_SQL_INIT**=***path_to_file* export *DBNAME*_SQL_INIT

For example:

MYDB_SQL_INIT=/usr/george/mystartfile export MYDB_SQL_INIT

where:

DBNAME is the name of the database and **must** be in uppercase.

VMS:

define DBNAME_SQL_INIT *path_to_file*

For example:

DEFINE /PROCESS MYDB_SQL_INIT DUA1:[USER.GEORGE]MYSTART.FILE

where:

DBNAME is the name of the database and **must** be in uppercase.

The file contains lines in the following format:

```
set lockmode session where readlock = nolock \g
```

This file contains the SET statements. If you have several set statements, separate the statements with a semicolon (;). You must end the entire file with "\g".

II_SQL_INIT

This name applies to the SQL single-line Terminal Monitor only. II_SQL_INIT points to SET statements within a file that are executed whenever this Ingres environment variable/logical takes effect.

II_SQL_INIT affects users who connect to the SQL terminal monitor. If set globally, it affects all users. If set interactively or locally, it affects only this user.

Windows: Global setting is with ingsetenv, or locally in the user's environment.

UNIX: Global setting is with ingsetenv, or locally in the user's .login, .profile or .cshrc file

VMS: Global setting is with define/system, define/group, or locally with define/process in the user's login.com file.

Setting this environment variable/logical is equivalent to a user's executing \i *filename* in the SQL terminal monitor each time they connect to a database.

Example: Set II_SQL_INIT

To define this Ingres environment variable, use the following command syntax at the operating system prompt:

Windows:

```
SET II_SQL_INIT=path_to_file
```

UNIX:

C Shell:

```
setenv II_SQL_INIT path_to_file
```

For example:

```
setenv II_SQL_INIT /usr/george/mystartfile
```

Bourne Shell:

```
II_SQL_INIT=path_to_file export II_SQL_INIT
```

For example:

```
II_SQL_INIT=/usr/george/mystartfile export II_SQL_INIT
```

VMS:

```
define II_SQL_INIT path_to_file
```

For example:

```
DEFINE /PROCESS II_SQL_INIT -DUA1:[USER.GEORGE]MYSTART.FILE
```

The file contains lines in the following format:

```
set lockmode session where readlock = nolock \g
```

This file contains the SET statements. If you have several set statements, separate the statements with a semicolon (;). You must end the entire file with "\g".

ING_SET

This name applies to SQL, QUEL single-line terminal monitors as well as Ingres tools such as ABF, VIFRED, embedded SQL, QBF, IQUEL, ISQL, and so forth. ING_SET points to SET statements that are executed whenever this Ingres environment variable/logical takes effect.

ING_SET affects users who connect to an application, an Ingres tool, or a terminal monitor. If set globally, it affects all users. If set interactively or locally, it affects only *this* user.

Windows: Global setting is with ingsetenv, or locally in the user's environment.

UNIX: Global setting is with ingsetenv, or locally in the user's .login, .profile, or .cshrc file

VMS: Global setting is with define/system, define/group, or locally with define/process in the user's login.com file.

ING_SET is set to a string surrounded by quotes. The string must be 64 characters or less, or it is invalid. The string can contain either:

- One or more set statements totaling no more than 64 characters

- The word include followed by the full path name to a file containing any number of set statements

Example: Set ING_SET

To define this Ingres environment variable, use the following command syntax at the operating system prompt:

Windows:

```
SET ING_SET = set-statement{; set statement}
```

Examples:

```
SET ING_SET=set lockmode show
SET ING_SET=include C:\users\default\mystartfile
```

UNIX:

C Shell:

```
setenv ING_SET 'set-statement {; set-statement}'
```

or

```
setenv ING_SET 'include path_to_file'
```

Examples:

```
setenv ING_SET 'set lockmode show'
setenv ING_SET 'include /usr/george/mystartfile'
```

Bourne Shell:

```
ING_SET='set-statement {; set-statement}'
```

or:

```
ING_SET='include path_to_file' export ING_SET
```

Examples:

```
ING_SET='set lockmode show' export ING_SET
```

```
ING_SET='include /usr/george/mystartfile' export ING_SET
```

VMS:

```
define ING_SET "set-statement {;set-statement}"
```

or:

```
define ING_SET "include path_to_file"
```

Examples:

```
DEFINE /PROCESS ING_SET "SET LOCKMODE SHOW"

DEFINE /PROCESS ING_SET -
"INCLUDE DUA1:[USER.GEORGE]MYSTART.FILE"
```

For the include format, the file specified by *path_to_file* contains the set statements. If you have several set statements, separate the statements with a semicolon (;). Place each set statement on a separate line in the file. For example:

```
set autocommit on;
set lockmode session where readlock=nolock;
set result_structure cbtree
```

ING_SET_DBNAME

This name applies to both SQL and QUEL single-line terminal monitors as well as Ingres tools such as ABF, VIFRED, embedded SQL, QBF, IQUEL, ISQL, and so forth. ING_SET_*DBNAME* points to SET statements that are executed whenever this Ingres environment variable/logical takes effect.

ING_SET_*DBNAME* affects users who connect to the database specified by *DBNAME* through an application, an Ingres tool, or a terminal monitor. If set globally, it affects all users of *DBNAME*. If set interactively or locally, it affects only this user.

Windows: Global setting is with ingsetenv, or locally in the user's environment.

UNIX: Global setting is with ingsetenv, or locally in the user's .login, .profile or .cshrc file.

VMS: Global setting is with define**/**system, define/group, or locally with define/process in the user's login.com file.

ING_SET_*DBNAME* is set to a string surrounded by quotes. The string must be 64 characters or less, or it is invalid. The string can contain either:

- One or more set statements totaling no more than 64 characters
- The word include followed by the full path name to a file containing any number of set statements

Example: Set ING_SET_DBNAME

To define this Ingres environment variable/logical use the following command entered at the operating system prompt:

Windows:

```
SET ING_SET_DBNAME=set-statement {;set statement}
```

For example:

```
SET ING_SET_MYDB=set lockmode show
SET ING_SET_MYDB=include C:\users\default\mystartfile
```

UNIX:

C Shell:

```
setenv ING_SET_DBNAME 'set-statement {; set-statement}'
```

or:

```
setenv ING_SET_DBNAME 'include path_to_file'
```

For example:

```
setenv ING_SET_MYDB 'set lockmode show'
setenv ING_SET_MYDB 'include /usr/george/mystartfile'
```

Bourne Shell:

```
ING_SET_DBNAME='set-statement {; set-statement}'
```

or:

```
ING_SET_DBNAME='include path_to_file' export ING_SET_DBNAME
```

For example:

```
ING_SET_MYDB='set lockmode show'export ING_SET_MYDB
```

```
ING_SET_MYDB='include /usr/george/mystartfile'export ING_SET_MYDB
```

VMS:

```
define ING_SET_DBNAME "set-statement {; set-statement}"
```

or:

```
define ING_SET_DBNAME "include path_to_file"
```

For example:

```
DEFINE /PROCESS ING_SET_MYDB - "SET LOCKMODE SHOW"
```

```
DEFINE /PROCESS ING_SET_MYDB - "INCLUDE DUA1:[USER.GEORGE]MYSTART.FILE"
```

For the include format, the file specified by *path_to_file* contains the set statements. If you have several set statements, separate the statements with a semicolon (;). Place each set statement on a separate line in the file.

For example:

```
set autocommit on;
set lockmode session where readlock=nolock;
set result_structure cbtree
```

ING_SYSTEM_SET

This name applies to both SQL and QUEL single-line terminal monitors as well as Ingres tools such as ABF, VIFRED, embedded SQL, QBF, IQUEL, ISQL, and so forth. ING_SYSTEM_SET points to SET statements that are executed whenever this Ingres environment variable/logical takes effect.

ING_SYSTEM_SET affects all users who connect to a DBMS Server through an application, an Ingres tool, or terminal monitor. It is always effective globally and can be set only by a privileged user.

ING_SYSTEM_SET is set to a string surrounded by quotes. The string must be 64 characters or less, or it is invalid. The string can contain either:

- One or more set statements totaling no more than 64 characters
- The word include followed by the full path name to a file containing any number of set statements

Example: Set ING_SYSTEM_SET

To define this Ingres environment variable/logical use the following command entered at the operating system prompt:

Windows:

SET ING_SYSTEM_SET = *set-statement{;set-statement}*

For example:

SET ING_SYSTEM_SET = set autocommit on

UNIX:

C Shell:

setenv ING_SYSTEM_SET '*set-statement{; set-statement}*'

For example:

setenv ING_SYSTEM_SET 'set autocommit on'
setenv ING_SYSTEM_SET 'include \
 /usr/ingres/ourstartfile'

Bourne Shell:

ING_SYSTEM_SET='set-statement {; set-statement}'

or:

ING_SYSTEM_SET='include path_to_file'

export ING_SYSTEM_SET

For example:

ING_SYSTEM_SET='set autocommit on'

export ING_SYSTEM_SET

ING_SYSTEM_SET='include \ /usr/ingres/ourstartfile'

export ING_SYSTEM_SET

VMS:

define ING_SYSTEM_SET "*set-statement* {; *set-statement*}"

or:

define ING_SYSTEM_SET "include *path_to_file* "

For example:

```
DEFINE /SYSTEM ING_SYSTEM_SET -
"SET LOCKMODE SHOW"

DEFINE /SYSTEM ING_SYSTEM_SET -
"INCLUDE DUA1:[INGRES]OURSTART.FILE"
```

For the include format, the file specified by *path_to_file* contains the SET statements. If you have several SET statements, separate the statements with a semicolon (;). Place each SET statement on a separate line in the file.

For example:

```
set autocommit on;
set lockmode session where readlock=nolock;
set result_structure cbtree;
```

INIT_INGRES

This name applies to the QUEL single-line terminal monitor only. INIT_INGRES points to SET statements within a file that are executed whenever this Ingres environment variable/logical takes effect.

INIT_INGRES affects users who connect to the QUEL terminal monitor. If set globally, it affects all users. If set interactively or locally, it affects only *this* user.

Windows: Global setting is with ingsetenv, or locally in the user's environment.

UNIX: Global setting is with ingsetenv, or locally in the user's .login, .profile, or .cshrc file.

VMS: Global setting is with define/system, define/group, or locally with define/process in the user's login.com file.

Setting INIT_INGRES variable/logical is equivalent to a user's executing \i *filename* in the QUEL terminal monitor each time they connect to a database.

Example: Set INIT_INGRES

To define this Ingres environment variable/logical use the following command entered at the operating system prompt:

Windows:

```
SET INIT_INGRES=path_to_file
```

For example:

```
SET INIT_INGRES= c:\user\mystartfile
```

UNIX:

C Shell:

```
setenv INIT_INGRES path_to_file
```

For example:

```
setenv INIT_INGRES /usr/george/mystartfile
```

Bourne Shell:

```
INIT_INGRES=path_to_file export INIT_INGRES
```

For example:

```
INIT_INGRES=/usr/george/mystartfile export INIT_INGRES
```

VMS:

```
define INIT_INGRES path_to_file
```

For example:

```
DEFINE /PROCESS INIT_INGRES -   DUA1:[USER.GEORGE]MYSTART.FILE
```

The file contains lines in the following format:

```
set lockmode session where readlock = nolock \g
```

This file contains the SET statements. If you have several set statements, separate the statements with a semicolon (;). You must end the entire file with "\g".

Startup

This name affects the QUEL single-line terminal monitor only.

The Ingres environment variable/logical startup affects all users of the QUEL terminal monitor. It is a customizable file (delivered with some macros already defined). It is located at: II_SYSTEM/ingres/files/startup.

Users with permission to edit this file can add statements of the such as the following:

```
{DEFINE RET; RETRIEVE}
```

The result of using this macro startup file is that users typing "RET TABLE1.ALL" has the string "RET" substituted by the string "RETRIEVE" at query execution time. For an explanation of how macros are used and defined, see the *QUEL Reference Guide.*

The quantity of some operating system resources available to each user session is fixed at Ingres DBMS Server start up. These include resources such as the quantities of memory available for parsing, storing and optimizing a query. Default values are used for these resources at installation time. These do not need to be changed unless you are receiving explicit error messages that indicate that these resources are inadequate.

Startsql

This name affects the SQL single-line terminal monitor only.

Startsql affects all users of the SQL terminal monitor. It is a customizable file located at: II_SYSTEM/ingres/files/startsql.

Users with permission to edit this file can add statements of the following example:

```
set lockmode session where readlock = nolock \g
```

Installation Locations

The values for the following Ingres variables are established at the time of installation:

- II_SYSTEM
- II_DATABASE
- II_CHECKPOINT
- II_JOURNAL
- II_DUMP
- II_WORK
- II_TEMPLATE
- ING_ABFDIR

Some of these values can be easily changed, and some require unloading of the installation.

When Not to Change Installation Locations

Before changing an existing Ingres installation location, be sure that the change is really necessary: ask yourself what needs to be moved, and why.

Installation locations do not need to be changed if:

- The installation needs space for future growth.
- Existing databases are outgrowing their present disk partitions.

Allocation of Space for Future Growth

Ingres provides the capability of creating new locations and placing any new databases, tables, or indexes in these new locations. If the requirement is space for **new** tables or databases, you do not need to change existing locations.

Instead, create the necessary new locations for these future tables and databases and use the available location flags of createdb (-d, -c, -j, -b, and -w) to create the data, checkpoint, journal, dump, and work files for the newly created database in some other location.

For more information on creating new databases, see the createdb command in the *Command Reference Guide*.

Allocation of Additional Space for Existing Tables

If space is becoming limited on a disk partition containing a particularly large table, you can move the table to a new disk or partition. Use the modify command to relocate and the MODIFY...TO REORGANIZE statements as described in the *Database Administrator Guide.*

Location Variables that Do and Do Not Require Unloading (UNIX and Windows)

When you need to change the value of an installation location variable, the procedure depends to some extent on the underlying operating system. On UNIX and Windows, the procedure also depends on whether the change requires unloading the database.

You can change the ING_ABFDIR location value without unloading the installation.

The values of the following locations are stored in the configuration file "aaaaaaaa.cnf" of each database at the time it is created and can only be changed by unloading each database, re-installing Ingres and reloading the databases (thus creating a new configuration file):

- II_SYSTEM
- II_DATABASE
- II_CHECKPOINT
- II_JOURNAL
- II_DUMP
- II_WORK

Change the Value of the ING_ABFDIR Variable (UNIX and Windows)

To change the installation default for ING_ABFDIR:

1. Shut down Ingres with ingstop
2. Change the value of ING_ABFDIR with the ingsetenv command
3. Restart the installation with ingstart

Note: Application developers who do not want to use the installation default can redefine ING_ABFDIR in their local user environment.

Change the Value of the ING_ABFDIR Logical (VMS)

To change the installation default value for ING_ABFDIR in VMS:

1. Shut down Ingres with ingstop.

2. Change the value with a new define command.

3. Restart the installation with ingstart.

ING_ABFDIR can be redefined in the local user environment for application developers who do not want to use the installation default.

Change an Installation Location with Unloading (Windows)

Follow these steps to change the value of II_SYSTEM, II_DATABASE, II_CHECKPOINT, II_JOURNAL, II_DUMP or II_WORK. "II_SYSTEM" is used as the example location to be changed:

1. Unload each database. For details, see the *Database Administrator Guide*.

2. Remove the old installation files by removing everything under the old %II_SYSTEM%\ingres directory and any extended locations.

3. Reinstall Ingres specifying the new value of II_SYSTEM.

4. Recreate each of the databases. For details, see the *Database Administrator Guide*.

    - Use the createdb *dbname* command to recreate a database.

    - Reload each database by appropriately editing and running the reload script.

5. Once you have reinstalled Ingres and recreated the databases in the new location, take new checkpoints of each of the databases. Checkpoints taken in old locations are no longer valid. For details, see the *Database Administrator Guide*.

Change an Installation Location with Unloading (UNIX)

Follow these steps in UNIX to change the value of II_SYSTEM, II_DATABASE, II_CHECKPOINT, II_JOURNAL II_DUMP, or II_WORK. "II_SYSTEM" is used as the example location being changed:

1. Unload each database. For details, see the *Database Administrator Guide.*

2. Remove the old installation files by removing everything under the old II_SYSTEM/ingres directory and any extended locations.

3. Set the value of II_SYSTEM and PATH to the desired new values, just as if you were installing Ingres for the first time.

4. Reinstall Ingres.

   Follow the instructions in the *Installation Guide* for a new (not upgrade) installation.

   As part of the installation procedure, you are asked to run the program ingbuild. This program prompts you for the new values of Ingres location variables II_DATABASE, II_CHECKPOINT, II_JOURNAL, II_DUMP, and II_WORK.

5. Recreate each of the databases. For details, see the *Database Administrator Guide.*

   Use the createdb *dbname* command to recreate a database.

   Reload each database by appropriately editing and running the reload.ing script.

6. Once you have reinstalled Ingres and recreated the databases in the new location, take new checkpoints of each of the databases. Checkpoints taken in old locations are no longer valid. For details, see the *Database Administrator Guide*.

Change Location Variables (Other than ING_ABFDIR) on VMS

Follow these steps in VMS to change the value of II_SYSTEM, user areas, II_DATABASE, II_CHECKPOINT, II_JOURNAL, II_DUMP, or II_WORK:

1. Shut down Ingres with ingstop.

2. Change the value of II_SYSTEM, II_DATABASE, II_CHECKPOINT, II_JOURNAL, II_DUMP, II_WORK, or user areas with the define command. Remember to modify any startup or other procedures where these logicals are defined.

3. Restart the installation with ingstart**.**

For more information about stopping and starting the installation using ingstop and ingstart, see the *Command Reference Guide.*

Use of Symbolic Links (UNIX)

Ingres does not support or recommend the use of symbolic links to change locations in UNIX. (This is possible with the UNIX ln command, which allows the UNIX system administrator to create symbolic links to new directories so that files are found in another directory even though the values of Ingres installation location variables remain unchanged.)

Common problems that arise from the misuse of symbolic links are as follows:

- Any files to which Ingres writes must reside on a file system that is local to the machine on which the writing process is running. This is necessary because of the operating system's inability to guarantee network writes. If symbolic links are used, writable directories can mistakenly be mounted from remote machines. This can cause write errors and data corruption.

- Occasionally Ingres executables reside on NFS file systems that have been exported with the - nosuid option. The permissions appear normal when the ls -l command is issued, but setuid programs fails with permission errors.

- The rm command does not remove files that are symbolic links. Instead, the symbolic link itself is broken, but the original file remains. This can cause cleanup routines to fail in some circumstances.

Chapter 5: Customizing Ingres

This section contains the following topics:

Ingres Character Sets

Ingres supports read-in character sets. This feature allows you to use a character set other than the default character set at your installation.

For example, in Thailand, the standard character set is Thai TIS. If you specify the THAI character set during the installation procedure, users at your installation will save and retrieve data using this character set.

The installation utility prompts you with a list of currently supported character sets for your platform. It sets the environment variable or logical for II_CHARSETxx to the value you have selected, unless the default character set is chosen. If the default character set is chosen, II_CHARSETxx is not set.

Important! *Choose your character set carefully. An installation can use only one character set. Do not change the character set because data in existing databases can be corrupted.*

After running the install program, check the value of II_CHARSETxx. If II_CHARSETxx is set incorrectly, you must run the installation program again.

For a complete list of supported character sets, see the *Installation Guide*. For a list of the character sets available on your platform, see the online list selection during installation.

Default Character Sets

The default character sets are as follows:

Windows: WIN1252

UNIX: ISO-8859/1

VMS: DEC Multinational

DEC Multinational and ISO-8859/1 are extended ASCII character sets. They have identical definitions for the 128 7-bit characters. They differ only in the defaults they assign for several of the 128 additional characters in the 8-bit character set.

Local Collation Sequences

A collation sequence determines the sorting order of any given character set. For example, in English, the order of the alphabet is most commonly used to order a list of English words. That is, words beginning with the letter c appear before words beginning with the letter d, which appear before words beginning with the letter e and so on.

A collation sequence is associated with each database when the database is created. This sequence determines in what order sorted data is returned to users and applications, what is returned when queries use pattern matching and, in some instances, how data is stored internally.

Supported Collation Sequences

In a computer, if no other collation sequence is enforced, the sequence derived from the machine's native character set—either ASCII or EBCDIC—is used. The sorting order of these character sets derives from the internal numeric representation of each character.

In addition to the sequences derived from ASCII and EBCDIC, Ingres supports the following local collation sequences:

- Multi, a sequence derived from the DEC Multinational Character Set
- Spanish, a sequence derived from the Spanish language
- Unicode_french, a French Unicode collation sequence for Unicode columns

If none of these sequences adequately fills your needs, you can write your own local collation sequence.

Multi Collation Sequence

The multi collation sequence is based on the DEC Multinational Character Set. This character set adds several vowels with diacritical marks to the standard 7-bit ASCII character set.

Following are the comparison sequences for the multi sequence that differ from those of ASCII:

A < À < Á < Â < Ã < Ä < B

C < Ç < D < E < È < É < Ê < Ë

I < Ì < Í < Î < Ï < J

N < Ñ < O

O < Ò < Ó < Ô < Õ < Ö < Œ < P

U < Ù < Ú < Û < Ü < V

Y < Ÿ < Z < Æ < Ø < Å

a < à < á < â < ã < ä < b

c < ç < d < e < è < é < ê < ë

i < ì < í < î < ï < j

n < ñ < o

o < ò < ó < ô < õ < ö < œ < p

ss < ß < st

u < ù < ú < û < ü < v

y < ÿ < z < æ < ø < å

For example:

```
cote < côte < czar < cæsar
```

Pattern matching rules:

- Œ, œ, ß are special when pattern match searching is used.
- Œ matches O_ or O% as well as Œ
- œ matches o_ or o% as well as œ
- ß matches s_ or s% as well as ß

Spanish Collation Sequence

The Spanish collation sequence is based on the multi sequence but contains additional support for the Spanish letters ll and ch. Listed below are the comparison sequences for the Spanish collation sequence that differ from those of ASCII. Some pattern matching rules are also described.

A < À < Á <Â < Ã < Ä < B

CZ < CÅ < Ç < CH < Ch < D < E < È < É < Ê < Ë

I < Ì < Í < Î < Ï < J

LZ < LÅ < LL < LI < M

N < Ñ < O

O < Ò < Ó< Ô < Õ < Ö < Œ < P

U < Ù < Ú < Û < Ü <V

Y < Ÿ < Z <Æ< Ø < Å

a < à < á < â < ã < ä < b

cz < câ < ç < cH < ch < d < e < è < é < ê < ë

i < ì < í < î < ï < j

lz < lå < lL < ll < m

n < ñ < o

o < ò < ó < ô < õ < ö < œ < p

ss < ß < st

u < ù < ú < û < ü < v

y < ÿ < z < æ < ø < å

Examples**:**

```
loop < llama
```

```
cote < côte < czar < cæsar < chair
```

The pattern matching rules are:

- Œ, œ, ß, Ç, ç are special when pattern match searching is used.
- Œ matches O_ or O% as well as Œ
- œ matches o_ or o% as well as œ
- ß matches s_ or s% as well as ß
- Ç matches C_ or C% as well as Ç
- ç matches c_ or c% as well as ç

Unicode_french Collation Sequence

The unicode_french collation is applicable to Unicode columns. It follows French sorting order for Unicode character strings where the accented characters are sorted from left to right instead of from right to left.

For example, in the default Unicode collation, the following is the sort order for these four Unicode strings:

cote

coté

côte

côté

The string starting with co.. comes before cô...

In unicode_french collation, the Unicode strings are sorted in the following order:

cote

côte

coté

côté

The strings ending in ..e are sorted before strings ending in ..é.

You can create a Unicode enabled database with unicode_french collation as follows:

```
createdb -iunicode_french unidb
```

or

```
createdb -nunicode_french unidb
```

Custom Collation Sequence

If you have special needs that are not met by the available collation sequences, you can write your own. Ingres allows you to write a collation sequence that has any of the following characteristics:

- Character skipping—one or more specified characters are ignored for collation

- One-to-one mapping—a character can be substituted for another or weighted differently for collation

- Many-to-one mapping—groups of characters can be substituted for a single character or weight value for collation

- Many-to-many mapping—groups of characters can be substituted for a sequence of characters or weight values for collation

Guidelines for Creating a Custom Collation File

Keep the following points in mind as you design and test your custom collation file:

- Never create a production database with an untested collation sequence. Always test your collation file on a sample database. Each time that you modify the collation sequence to correct any bugs, you must unload the database, destroy the old database, install the new sequence, create a database with the new sequence, and reload the database.

- Some collation sequences allow two strings that are different to compare as equal. These sequences are called information loss sequences. An example of this type of sequence is a sequence that ignores case. Problems that can result from such a sequence are:

  - If duplicates are not allowed, the DBMS drops all but one string.

  - If duplicates are not allowed, the DBMS does not allow you to add a row to a table if it appears to match an existing row.

  - The hash storage structure cannot detect when two equal but different strings are placed in a hashed relation.

  - In a query on a hash table, the "=" operator can only fetch one of the 'equal' strings that matches the expression.

Because of these problems, we suggest that you do not use information loss sequences and the hash storage structure together.

How You Write a Customized Collation Sequence

To create a customized collation sequence, follow these steps:

1. Write a description file.

2. Run the description file through the aducompile utility.

3. Test your collation sequence with a small sample database.

Description File—Describe Collation Sequence

To define a custom collation sequence you must create a description file, which consists of a list of "instructions" that, taken as a whole, describe the collation sequence. Each instruction must appear on a separate line in the file.

The format of each instruction is:

value:*string*

where:

value

Determines the numerical weight assigned to string. (The internal numerical weight of each character determines where a character appears in the sort order.)

The *value* can have any of the following formats:

char+number

Instructs sorting of the specified string after the specified character and before the next higher-weighted character in the character set. For example, in the following instruction, *string1* is mapped as a single character that is ordered immediately after the letter H and before I in a sorted sequence:

```
H+1:string1
```

In the following instruction, *string2* sorts after *string1* and before the letter I.

```
H+2:string2
```

You can specify H+1:*string* or Hz+1:*string* and both sorts in the same manner, that is, after H and before I. However, the two examples do not behave the same when pattern matching is applied. To illustrate using an example from the Spanish language, the following instruction maps CH as a single character that exists between C and D:

```
C+1:CH
```

If you ask for a pattern match using the format C%, instances of CH are not returned. The alternative, Cz+1:CH, maps CH into two characters, C and a virtual character just after z. This causes CH to match as two characters. A pattern match using the format C% finds the instances of CH.

charstring

Sorts the specified string as the equivalent of the specified charstring. For example, in the following instruction, the word tax sorts as if it were the word revenue:

```
revenue:tax
```

+number

Gives the specified string the internal numerical weight specified by given number. The number must be between 0 and 32766. The weighting of a character in this manner is less portable than giving the character a relative weight.

+*

Causes the specified string to be ignored when collation is performed. For example, in the following instruction, the "?"is ignored whenever collation takes place.

```
+*:?
```

(empty)

When no value is specified (the instruction takes the form: string), the collation compiler ignores the instruction. Use this format to insert comments into your collation sequence. For example:

```
:This is a comment
```

string

Is any character or character string. An empty string causes a syntax error.

The aducompile Utility

The aducompile utility compiles the description file for your collation sequence into a binary file and installs that file as a collation sequence that can be used. You must be the installation owner to use this utility. Be sure to give your resulting collation file a unique name so that you do not overwrite any existing collation files.

Your new collation sequence is located at $II_SYSTEM/ingres/files/collation/*collation_name.*

Note: In UNIX, all system users must have rights to read the new collation file.

Archiver Exit Script (acpexit)

The Archiver Process Exit command script is executed by the archiver before an error halt. This shell script (acpexit in UNIX or ACPEXIT.COM in VMS) resides in the files directory. You can use the default script behavior, which sends mail to the installation owner account with the reason for the archiver failure, or you can edit the script to customize certain aspects of archiver recovery.

Because the archiver writes sequentially to the end of journal files, a common problem occurs when the archiver stops as a result of running out of journal disk space. If left unresolved, running an installation without an archiver (and with journaled databases) eventually causes the log file to fill.

To address this concern, whenever the archiver detects an error condition that forces it to stop (including running out of journal disk space) it first executes the archiver exit script.

Customization of Archiver Exit Script

You can modify the archiver exit script to take action appropriate to your installation. For example, one possible course of action is to delete a reserved file on the journal disk to free up space and restart the archiver.

You can customize the archiver exit script in these respects:

- The default script sends mail to the installation owner account when archive processing halts, and includes the reason for the failure. You can modify this script to notify the system administrator directly of a situation that needs resolving.

- You can tailor the script to handle certain resource and environmental problems automatically. For example, if archive processing has stopped because of insufficient disk space, the script can automatically reclaim disk space and resume archive processing.

- The default script is executed with one or two arguments, described in Archiver Exit Script Parameters (see page 92). You can choose whether to incorporate these arguments if you customize the script.

The default script includes examples of processing you can use for your installation. These examples are formatted as comments, and can be included in the script by removing the comment prefixes. The script is located at $II_SYSTEM/ingres/files/acpexit.

Archiver Exit Script Parameters

The archiver exit script parameters are as follows:

Parameter	Description	Status
errornumber	An error number that describes the reason archive processing has stopped. At the end of the default script is a list of all possible error numbers, with an explanation of the error and, in many cases, the recommended actions to correct the error.	Always passed in default script.
database_name	The name of the database that was being processed at the time archiving stopped. The name is provided only if the halt in archive processing was associated with a particular database problem, as opposed to a general system problem.	Sometimes passed in default script.

How to Use a Custom Mapping File for Unicode Coercion

The DBMS Server coerces Unicode data to local character set value based on the Unicode converter (or mapping table). Ingres automatically chooses the converter to use, based on the encoding used by the locale on your system.

You can create a custom mapping table for coercing Unicode data to local character set value if, for example, you want to map a particular character to a different Unicode code point, or create a completely new mapping table for an encoding not yet supported by Ingres. Always use the custom mapping table for your encoding once you have implemented it.

Always create a new mapping file. If you wish to modify an originally supplied mapping table for an encoding, do not modify the original mapping files but copy them to a new file name. To avoid data corruption, test your mapping before implementing in a production installation.

If you are using a custom converter (mapping table) for an encoding, make sure that you always use the same converter for that encoding.

The process for creating and using a custom mapping file is as follows:

1. Create a custom mapping file in XML format.

    This format should conform to charactermapping.dtd present in the directory $II_SYSTEM/ingres/files/ucharmaps. This directory also contains other converters, which may serve as examples of xml file format. For more information on the xml file format, see http://www.unicode.org/reports/tr22/.

2. Compile this mapping file using unimapcompile utility, as follows:

    ```
    unimapcompile -o=yourmapfilename yourmapfilename.xml
    ```

3. Set the mapping file in the symbol table for the server using the ingsetenv command, as follows:

    ```
    ingsetenv II_UNICODE_CONVERTER yourmapfilename
    ```

    To check that the value is set correctly, run:

    ```
    ingprenv II_UNICODE_CONVERTER
    ```

4. If you have remote clients connecting to this server and will be using the encoding for which you have created the custom mapping table, then copy the compiled mapping file to all clients in the directory $II_SYSTEM\ingres\files\ucharmaps. Also ensure that you have set II_UNICODE_CONVERTER to your custom converter in the symbol table for the clients.

Chapter 6: Troubleshooting Ingres

This section contains the following topics:

The procedures in this chapter will help you to solve problems in your Ingres installation or to accurately define the problem if it is still necessary to call technical support.

Process of Troubleshooting

Troubleshooting is a process of defining and correcting a problem that occurs in an otherwise functioning Ingres installation and includes narrowing down the problem to a well-defined point, identifying the cause of failure, and eliminating it. When troubleshooting, you:

1. Determine the nature of the problem

2. Isolate the problem to a defined area

3. Eliminate the cause, using the following techniques:

   - Correcting user errors

   - Changing the Ingres installation environment

   - Changing the user environment

   - Changing the operating system environment

   - Changing the system configuration

   - Restarting the system if needed

Troubleshooting Tools

VDBA provides a set of system administration tools. It allows you to perform system administrator functions—including configuring, performance monitoring, backup and recovery, and remote database optimization—using a GUI interface. For instructions on performing these functions, see the VDBA online help.

Command line instructions are also available. For information on Ingres commands, see the *Command Reference Guide*.

Determine the Problem Area

The first step in the process of troubleshooting is to determine the problem area. The following troubleshooting flow chart shows the major problem categories:
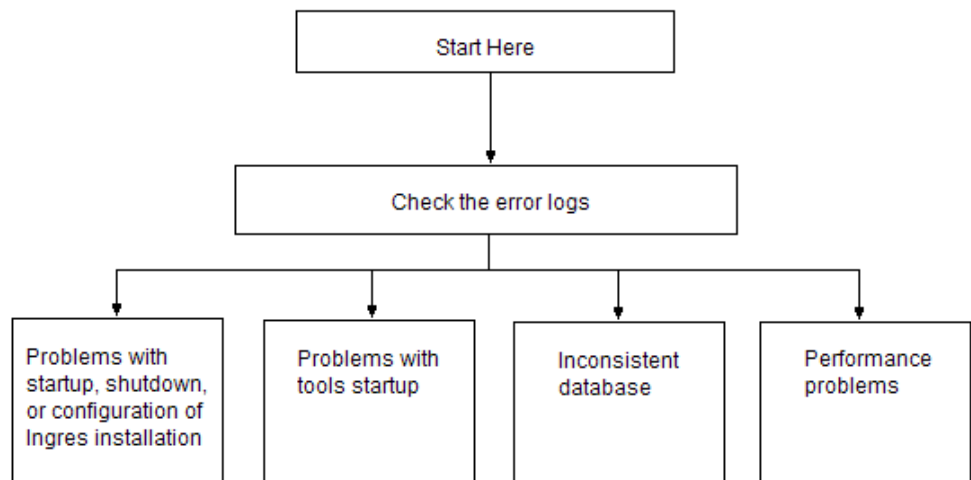
Error Log Files

The error log files are located in the directory indicated by the Ingres environment variable/logical II_CONFIG. The log files are as follows:

errlog.log

> The main DBMS error log and the default log for most programs.

iiacp.log

> Archiver error log.

iircp.log

> Recovery error log.

Note: If your system is configured for Ingres Cluster Solution, each node in the Ingres cluster maintains a separate Archiver and Recovery error log. Each log is distinguished by having _nodename appended to the base log name, where *nodename* is the Ingres node name for the host machine as returned by iipmhost.

On VMS, if SCSNODE is set, the log files except errlog.log is named *.LOG_*nodename*.

Optional Error Log Files

The names of optional log files can vary, but all log files end with the extension LOG. Optional log files include:

II_DBMS_LOG

> DBMS error log

II_GC_LOG

> GCC trace log

II_STAR_LOG

> Star error log

Display Value for II_CONFIG

To display the value for II_CONFIG, type the following command at the operating system prompt:

Windows:

```
ingprenv
```

UNIX:

```
ingprenv
```

VMS:

```
set def II_SYSTEM:[INGRES.FILES]
```

or

```
set def II_CONFIG.
```

View List of Log Files

For a list of log files, type the following command at the operating system prompt from the directory indicated by $II_CONFIG (UNIX) or II_CONFIG (VMS):

Windows:

```
dir *.log
```

UNIX:

```
ls *.log
```

VMS:

```
dir *.log
```

Check the Error Log Files

Checking for error logs is the first step for determining the nature of the problem. To check for error logs, follow these steps:

1. Check the log files for error messages. Examine errlog.log first.

2. If indicated, check optional log files for error messages.

3. Identify the errors associated with your problem. All errors are time stamped. Find the most recent error message associated with your problem. Read back up the log file from there to the first error relating to that problem. Many error messages cascade from that initial error. This initial error is usually the most important in identifying the problem even though it is not the error that users report.

   For example, the following error message merely notifies you that a DBMS server has exited for one of many possible reasons:

   E_SC0221_SERVER_ERROR_MAX

   You must look in the errlog.log for associated DBMS server errors such as "E_DM9300_DM0P_CLOSE_PAGE Buffer still fixed," a fatal error message. Search the error log for additional errors occurring around the time of the error that was displayed.

Find Your Problem Category

To find the problem category, follow these steps:

1. Determine the general category in which your problem lies. The following categories relating to running the Ingres installation are described in this guide:

   - Ingres startup and shutdown

   - Ingres configuration

   - Ingres tools startup

   - Inconsistent database or recovery

   - Operating system performance

   For a description of problems relating to queries and performance, see the *Database Administrator Guide.*

2. Use the information in the error logs to determine which category to check. Error messages always include the first two letters of the facility code that generated the error.

Troubleshoot Startup, Shutdown, or Configuration Problems

Use the following flow chart to isolate a problem with startup, shutdown or configuration of your Ingres installation:

```
┌──────────────────┐                  ┌───────────────────────────┐
│ Did failure occur│                  │ See section on Detect Ingres Startup │
│ immediately after│ ──Yes──▶         │ Problems on System Reboot. *         │
│ machine reboot?  │                  └───────────────────────────┘
└──────────────────┘
        │
        No
        │
        ▼
┌──────────────────┐                  ┌───────────────────────────┐
│ Are you running  │ ──Yes──▶         │ See section on Check Shutdown        │
│ ingstop?         │                  │ Problems. *                          │
└──────────────────┘                  └───────────────────────────┘
        │
        No
        │
        ▼
┌──────────────────┐                  ┌─────────────────────────────────────┐
│ Are you running  │ ──Yes──▶         │ Does startup proceed as far as starting │
│ ingbuild?        │                  │ the installation processes (did you see a │
└──────────────────┘                  │ message indicating that the first process │
                                      │ has started)?                         │
                                      └─────────────────────────────────────┘
                                                   │
                                          ──No──▶  ┌──────────────────────┐
                                          Yes      │ See ingbuild section.│
                                                   └──────────────────────┘
        No
                                                   ▼
                                      ┌─────────────────────────────────────┐
                                      │ See section on Detect Ingres Startup │
                                      │ Problems. *                          │
                                      └─────────────────────────────────────┘
        ▼
┌──────────────────┐                  ┌──────────────────────────┐
│ Do all of the Ingres │ ──Yes──▶     │ Are you able to connect to a │
│ processes start up?  │              │ database?                    │
└──────────────────┘                  └──────────────────────────┘
        │                                        │
        │                              No────Yes────▶ ┌──────────────┐
        No                                            │ Done! Problem│
        │                                             │ solved!      │
        │                                             └──────────────┘
        │                                        ▼
        │                             ┌──────────────────────────────────┐
        │                             │ See section on Problems with Tools Startup. │
        │                             └──────────────────────────────────┘
        ▼
```

```
┌─────────────────┐                  ┌──────────────────────────────────┐
│ Is the recovery │─── No ──────────▶│ See section on Check for Recovery │
│ process running?│                  │         Process Errors. *          │
└─────────────────┘                  └──────────────────────────────────┘
        │
       Yes
        ▼
┌─────────────────┐                  ┌──────────────────────────────────┐
│ Is the archiver │─── No ──────────▶│ See section on Check for Archiver │
│ process running?│                  │         Process Errors. *          │
└─────────────────┘                  └──────────────────────────────────┘
        │
       Yes
        ▼
┌─────────────────┐                  ┌──────────────────────────────────┐
│  Is the Name    │                  │ See section on Check for Name     │
│ Server running  │─── No ──────────▶│       Server Errors. *             │
│ and did it start│                  │                                    │
│ without errors? │                  └──────────────────────────────────┘
└─────────────────┘
        │
       Yes
        ▼
┌─────────────────┐                  ┌──────────────────────────────────┐
│    Is the       │─── No ──────────▶│    Did you install Ingres Net?     │
│ Communication   │                  └──────────────────────────────────┘
│ Server running? │                         │
└─────────────────┘                        Yes ─── No
        │                                   │
       Yes                                  ▼
        │                        ┌──────────────────────────────────┐
        │                        │ See section on Check for           │
        │                        │ Communication Server Process       │
        │                        │ Errors. *                          │
        │                        └──────────────────────────────────┘
        ▼
┌─────────────────┐                  ┌──────────────────────────────────┐
│  Is the DBMS    │                  │ See section on Check for DBMS      │
│ Server process  │─── No ──────────▶│       Server Process Errors. *     │
│    running?     │                  └──────────────────────────────────┘
└─────────────────┘
        │
       Yes
        ▼
┌─────────────────┐                  ┌──────────────────────────────────┐
│ Are you able to │                  │        Done! Problem Solved!       │
│  connect to a   │─── Yes ─────────▶│                                    │
│   database?     │                  └──────────────────────────────────┘
└─────────────────┘
        │
        No
        ▼
┌──────────────────────────────────────────────────────────────────────┐
│            See section on Problems with Tools Startup.                 │
└──────────────────────────────────────────────────────────────────────┘
```

* Note: Refer to the sections specific to the platform Ingres is running on.

Check Ingres Installation on Windows

To check if the Ingres installation is working fine, follow these steps:

1. Check that you are logged in as the installation owner. If not, log off and log in again as this user.

2. Check that all users have II_SYSTEM set by issuing the following command at the operating system prompt:

   `echo %II_SYSTEM%`

   All users must have Ingres executables in their path variables. Check that everyone has the full search path to %II_SYSTEM%\ingres\bin.

   The installation owner must also include %II_SYSTEM%\ingres\utility.

3. Check that each of the Ingres variables has a valid value.

   Ingres environment variables are only used and "seen" by Ingres and can be displayed with the following command entered at the operating system prompt:

   `ingprenv`

   If you are in doubt about the function or legal value of a particular environment variable, see the chapter "Setting Environment Variables and Logicals" and the appendix "Environment Variables and Logicals."

   Ingres environment variables denoting installation locations (II_DATABASE, II_CHECKPOINT, II_DUMP, II_JOURNAL, II_LOG_FILE, II_SYSTEM) cannot be reset. To change these, you must unload the databases, rerun the installation program, and reload the databases, as described in Installation Location (see page 77).

4. Check the Ingres environment variables that have been set locally, overriding the Ingres installation-level definitions.

   Only a small category of Ingres environment variables must be defined in the local user environment: those that permit you to access Ingres, and those that define values that are different for your local environment. They include TERM_INGRES and ING_EDIT.

   If you trace the problem to an Ingres environment variable setting, correct the value. For procedures and scenarios for setting these environment variables, see the chapter "Setting Environment Variables and Logicals." If the installation does not start up, continue with this procedure.

5. Check that all Ingres processes are shut down. If there are processes that continue to run, see the section Check Shutdown Problems on Windows (see page 114).

6. Restart Ingres. Attempt once again to start up the installation by issuing the following command at the operating system prompt:

   `ingstart`

If startup problems persist, continue diagnostics in the ingstart section below.

Detect Ingres Startup Problems on Windows

To troubleshoot startup problems on Windows, follow these steps:

1. Determine which Ingres processes are running.

2. Verify that all required Ingres system processes are running. The following processes are the minimum required for a complete installation:

   iigcn

   > Name Server process

   iigcc

   > Communication Server process (present only on sites with Ingres Net)

   II_IUSV_*nnn*

   > Recovery Server process

   dmfacp

   > Archiver process

   iidbms

   > DBMS Server process

   iigcd

   > Data Access Server process

   iistar

   > Star Server process (present only on sites with Ingres Star)

   rmcmd

   > Remote command process

   iigcb

   > Bridge Server process (present only on sites with Ingres Bridge)

3. If ingstart does not complete successfully, do the following:

   a. Check the output from ingstart. This is saved in the file:

      ```
      %II_SYSTEM%\ingres\files\ingstart.log
      ```

      Error log files and individual component log files are listed in Logging and Locking Systems (see page 29).

   b. Try to determine the reason for startup failure.

   c. A process failed to start. If a process failed to start, continue on to the detailed sections on startup problems for that specific process.

Check Ingres Installation on UNIX

To check if the Ingres installation is working fine, follow these steps:

1. Check that you are logged in as the installation owner by issuing the following command at the operating system prompt:

   `whoami`

   If the user ID of the installation owner is not shown, log off and log in again as this user.

2. Check that all users have II_SYSTEM set by issuing the following command at the operating system prompt:

   `echo $II_SYSTEM`

   /usr/r6 (this varies by system)

   All users must have Ingres executables in their path variables. Check that everyone has the full search path to $II_SYSTEM/ingres/bin.

   The installation owner must also include $II_SYSTEM/ingres/utility.

3. Check that each of the Ingres installation variables has a valid value.

   Ingres environment variables are only used and "seen" by Ingres and can be displayed with the following command entered at the operating system prompt:

   ingprenv

   If you are in doubt about the function or legal value of a particular environment variable, see the chapter "Setting Environment Variables and Logicals" and the appendix "Environment Variables and Logicals."

   Ingres environment variables denoting installation locations (II_DATABASE, II_CHECKPOINT, II_DUMP, II_JOURNAL, II_LOG_FILE, II_SYSTEM) cannot be reset. To change these, you must rerun the installation program, ingbuild, and on UNIX possibly unload and reload your database with unloaddb. More information is provided in Installation Locations (see page 77).

4. Check the Ingres environment variables that have been set locally, overriding the Ingres installation-level definitions. Issue the following commands at the operating system prompt:

BSD:

```
printenv | grep II
```

```
printenv | grep ING
```

System V:

```
env | grep II
```

```
env | grep ING
```

Only a small category of Ingres environment variables must be defined in the local user environment: those that permit you to access Ingres, and those that define values that are different for your local environment. They include TERM_INGRES and ING_EDIT.

If you trace the problem to an Ingres environment variable setting, correct the value. For procedures and scenarios for setting these environment variables, see "Setting Environment Variables and Logicals." If the installation does not start up, continue with this procedure.

5. Identify your installation code. If there is more than one Ingres installation on this machine, type the following command at the operating system prompt. The installation code is used to distinguish which processes belong to which installation at sites with more than one Ingres installation on the same machine:

```
ingprenv | grep II_INSTALLATION
```

The two-letter installation code is displayed (for example, the following code R6):

```
II_INSTALLATION=R6
```

Take note of your installation code: _____.

6. Check that all Ingres processes are shut down. If there are processes that continue to run, see Check Shutdown Problems on UNIX (see page 116).

7. Restart Ingres: Attempt once again to start up the installation by issuing the following command at the operating system prompt:

```
ingstart
```

8. If startup problems persist, continue the diagnostics described in Ingbuild on UNIX (see page 106) or Detect Ingres Startup Problems on UNIX (see page 107).

Ingbuild on UNIX

The executable script ingbuild performs all the steps necessary to set up an installation. It checks system resources, installs shared memory and semaphores, configures DBMS server parameters, configures the logging and locking system, and starts all the required processes.

The ingbuild program is located in $II_SYSTEM/ingres/utility. It makes use of numerous shell commands as well as the following Ingres binary and shell executables:

- createdb

- iilink

- ingstop

- ingstart

- ingprenv

- ingunset

- sql

One of the last things ingbuild does is call the ingstart script to actually start installation processes. When ingstart is called, it displays the message "Starting the Name Server process (iigcn)." If there are startup problems after this message has displayed, see Ingstart on UNIX (see page 107).

Before you can diagnose a problem with ingbuild, you must identify which subroutine is failing. If you know which routine is failing and it is ingstart or one of the main installation processes (iigcn, iigcc, II_IUSV_nnn, dmfacp or iidbms), see the section below that addresses that executable.

Details on tracing are described in Bourne Shell -x Option (see page 211).

Detect Ingres Startup Problems on UNIX

To diagnose Ingres problems, use the following procedure.

1. Display which processes are running by using the csreport and operating system ps commands.

   The csreport utility is described in Operating System Utilities (see page 175) and the ps command is described in UNIX Operating System Utilities (see page 177).

2. Verify that all required Ingres system processes are running. The following processes (in the order they are started) are the minimum required for a complete installation:

   iigcn

   Name Server process

   iigcc

   Communications Server process (present only on sites with Ingres Net)

   iidbms (II_IUSV_nnn)

   Recovery Server process

   iigcd

   Data Access Server process

   dmfacp

   Archiver process

   iidbms

   DBMS Server process

   iislave

   DBMS Server asynchronous write daemons—these processes are created at the time of the first database write operation (only present on sites where Ingres does not use OS threads or asynchronous I/O)

   iistar

   Star Server process (present only on sites with Ingres Star)

   iigcb

   Bridge Server process (present only on sites with Ingres Bridge)

   Note: If the command ingprenv | grep II_CLIENT shows "II_CLIENT = true", you need to run only the Name Server and Communications Server processes.

In a few special cases Ingres can be run without the Communications Server process. This includes releases that use the streams device driver for interprocess communications. If you are not sure if your release uses the streams device driver, see your Readme file.

3. If ingstart does not complete successfully, try to identify the reason for startup failure. For example:

   The problem is with ingstart. The ingstart script fails due to results of the checks it makes for sufficient resources and installation settings. If this is the reason for startup failure, correct the deficiency.

   A process failed to start. If a process failed to start, continue on to the details sections on startup problems for that specific process.

Check Ingres Installation on VMS

It is recommended that you first make the following basic installation checks:

1. Check that you are logged in as the ingres system administrator (ISA) user by issuing the following command at the operating system prompt:

   ```
   show process
   ```

   If the ISA is not shown, log off and log in again as the ISA. The ISA is typically ingres if there is only a single installation per machine. For multiple installations, there is a different ISA for each installation.

2. Check that all users have II_SYSTEM set by issuing the following command at the operating system prompt:

   ```
   show logical II_SYSTEM
   ```

3. Check that each of the Ingres installation logicals has a valid value.

   Ingres symbols are defined in the following files:

   - II_SYSTEM:[INGRES.UTILITY]INGSYSDEF.COM - Defines Ingres system administration, DBA, and user symbols.

   - II_SYSTEM:[INGRES.UTILITY]INGDBADEF.COM - Defines Ingres DBA and user symbols.

   - II_SYSTEM:[INGRES.UTILITY]INGUSRDEF.COM - Defines Ingres user symbols.

   If you are in doubt about the function or legal value of a particular logical, see the chapter "Setting Environment Variables and Logicals" and the appendix "Environment Variables and Logicals".

Ingres logicals denoting installation locations (II_DATABASE, II_CHECKPOINT, II_DUMP, II_JOURNAL, II_LOG_FILE, II_SYSTEM) can not be reset. To change these, you must edit II_CONFIG:config.dat, update the existing definition, and execute @ii_system:[ingres]ingsysdef.com. For additional details, see Installation Locations (see page 77).

4. Check the Ingres logicals that have been set locally, overriding the Ingres installation-level definitions. Issue the following command at the operating system prompt:

   ```
   show logical *II*, *ING*
   ```

   This shows which ones have been set at process, job, or group level, overriding the system-level definitions.

   Only a small category of Ingres logicals are defined in the local user environment: those that permit you to access Ingres, and those that define values that are different for your local environment. They can include TERM_INGRES and ING_EDIT.

   If you trace the problem to an Ingres logical setting, correct the value. For procedures and scenarios for setting these logicals, see the chapter "Setting Environment Variables and Logicals." If the installation does not start up, continue with this procedure.

5. Identify your installation code if there is more than one Ingres installation on this machine. To do so, type the following command at the operating system prompt:

   ```
   show logical II_INSTALLATION
   ```

   The two-letter installation code is displayed. For example:
   ```
   II_INSTALLATION=R6
   ```

   If this is a system-wide Ingres installation, there is no definition for II_INSTALLATION. The installation code is used to distinguish which processes belong to which installation at sites with more than one Ingres installation on the same machine:

6. Check that all Ingres processes are shut down. If processes continue to run, see Check Shutdown Problems on VMS (see page 119).

7. Restart Ingres. Attempt once again to start up the installation by issuing the following command at the operating system prompt:

   ```
   ingstart
   ```

8. If startup problems persist, continue diagnostics contained in the next section—vmsinstal or ingstart.

VMSINSTAL on VMS

The executable script vmsinstal performs all the steps necessary to set up an installation. It checks system resources, installs shared images, configures DBMS server parameters, configures the logging and locking system, and starts all the required processes.

During the execution of the vmsinstal script, if a problem occurs, do the following:

1. Execute the vmsinstal program with the **L** option to log the contents of the VMSINSTAL session:

   ```
   @sys$update:vmsinstal * device_name options L
   ```

2. Examine the DCL by issuing the following command at the operating system prompt:

   ```
   set verify
   ```

3. Examine the vmsinstal log for error notifications and attempt to resolve the problem.

4. If the Installation Verification Procedure (IVP) for the DBMS Server fails with the message "ii.node_name.syscheck.gblpages not found in config.dat" after you enter the number of concurrent users that the installation supports, check the value of the account parameter PRCLM (process limit) to ensure that it is not zero. The PRCLM must be a minimum of 15.

5. If the problem persists, contact technical support, as described in What You Need Before Contacting Technical Support (see page 170).

Detect Ingres Startup Problems on VMS

If you are having ingstart problems, use the following procedure:

1. Display the processes that are running by issuing the following command at the operating system prompt:

```
show system /out = filename

search filename "II_", "DMF"
```

The search command searches the user-specified *filename* for the process names.

2. Verify that all required Ingres system processes are running. The following processes (in the order they are started) are the minimum required for a complete installation:

- II_IUSV_*nn_ppp*—Recovery process

- DMFACP_*nn*—Archiver process

- II_GCN_*nn*—Name Server process

- II_GCC_*nn_pppp*—Communications Server process (present only on sites with Ingres Net)

- II_DBMS_*nn_pppp*—DBMS Server process

- II_STAR_*nn_pppp*—Star Server process (present only on sites with Ingres Star)

- RMCMD_*nn_ppp*—Remote Command Server for use with VDBA

- II_GCB_*nn_pppp*—Bridge Server process (present only on sites with Ingres Bridge)

Where *nn* is the optional two-letter installation code, and *pppp* is the lower two bytes of the process' process ID.

Note: If you are checking the processes on a client node, only the Name Server and Communications Server processes are displayed when issuing the $ SHOW SYSTEM command.

3. If ingstart does not complete successfully, do the following:

   a. Check for error messages in the errlog.log:

   ```
   II_SYSTEM:[INGRES.FILES] II_CONFIG:errlog.log
   ```

   For a complete list of error log files and individual component log files, see Logging and Locking Systems (see page 29).

   b. Try to identify the reason for startup failure.

   - The problem is with ingstart. If the ingstart executable fails due to results of the checks it makes for sufficient resources and installation settings, correct the deficiency.

   - A process failed to start. If a process failed to start, continue on to the detail sections on startup problems for that specific process.

Detect Ingres Startup Problems on System Reboot (Windows)

To detect Ingres startup problems on system reboot on Windows, follow these steps:

1. Check that the "Startup Type" for the "Ingres Intelligent Database" service has been set to "automatic" in the services dialog.

2. Check that the password specified for the "Ingres Intelligent Database" service matches the installation owner user password.

3. Start the service manually.

4. If the service fails to start, run ingstart from the command prompt.

5. If the installation still does not start, contact technical support, as described in What You Need Before Contacting Technical Support (see page 170).

Detect Ingres Startup Problems on System Reboot (UNIX)

To detect Ingres startup problems on system reboot on UNIX, follow these steps:

1. The most common cause of startup failure following a reboot is failure to include the startup command ingstart in the boot script for your machine. (The boot file is vendor-specific but can be named "/etc/rc" or "etc/rc.local".) This file contains the commands that are to be executed immediately after a reboot.

   Make sure that the following line appears in the boot script:

   ```
   su userid -c "ii_system/ingres/utility/ingstart ii_system" /dev/console
   ```

   where:

   userid refers to the user that owns the installation

   ii_system refers to the value of II_SYSTEM for your installation.

2. Make sure that /dev/kmem is readable to the user that owns the installation. If this is a security problem for your machine, you can add this user as a member of /dev/kmem's group and make the /dev/kmem group readable.

   Issue the following command at the operating system prompt:

   ```
   chmod g+w /dev/kmem
   ```

   The user that owns the installation *must* be able to read /dev/kmem or the kernel resource checks in ingstart fails.

3. Run ingstart. If the installation still does not start, contact technical support, as described in What You Need Before Contacting Technical Support (see page 170).

Detect Ingres Startup Problems on System Reboot (VMS)

To detect Ingres startup problems on system reboot on VMS, follow these steps:

1. Make sure that a real error has occurred. If Ingres is performing startup recovery, see Automatic Recovery (see page 148).

2. Check to see that the Ingres initialization code is called from the VMS startup files. There are two main startup files:

   - SYS$MANAGER:SYLOGIN.COM

     SYLOGIN.COM is run when any user logs in. The script to set up Ingres user symbols are defined here.

     `@II_SYSTEM:[INGRES]ingusrdef.com`

     If the DBA symbols are also to be known by all users, the ingdbadef.com script can be used instead.

     If the system administrator symbols are also to be known by all users, the ingsysdef.com script can be used instead.

     Only one script needs to be defined.

     An alternative method is to include these in a user's particular login.com.

   - SYS$MANAGER:SYSTARTUP_VMS.COM

     SYSTARTUP_VMS.COM is run when the system is booted. To have Ingres restarted automatically on reboot, add the following line:

     `@SYS$STARTUP:INGRES_STARTUP.COM`

     Edit the file SYS$STARTUP:INGRES_STARTUP.COM to ensure it is correct for your environment. A line is added to start *each* installation, but the line is commented out to prevent accidental errors.

     Edit this file to meet your needs.

3. Be sure that you have examined all of the log files. Look in these files for possible error messages:

   - II_CONFIG:ERRLOG.LOG

   - II_CONFIG:IIACP.LOG (with _*nodename* name suffix if configured for clusters)

   - II_CONFIG:IIRCP.LOG (with _*nodename* name suffix if configured for clusters)

   - II_CONFIG:II_CSP.LOG_*nodename* (cluster installations only)

4. Determine if any system logicals have been changed interactively since the last startup of Ingres but the command files have not been updated (use commands such as define/system/exec). Logical names become undefined when VMS is shut down and must be redefined (using the command files) at each boot.

5. Make sure that DECnet and/or TCP/IP is started before either Ingres Cluster Option or Ingres Net is started.

6. Execute the ingstart utility to start up Ingres.

   If the installation still does not start, contact technical support, as described in What You Need Before Contacting Technical Support (see page 170).

Check Shutdown Problems on Windows

If Ingres has problems during shutdown on Windows, follow these steps:

1. Check environment variables in the local user environment by entering the following commands at the operating system prompt:

   a. Verify that II_SYSTEM is set correctly.

      ```
      echo %II_SYSTEM%
      ```

   b. Check that you have the full search path to %II_SYSTEM%\ingres\bin, and %II_SYSTEM%\ingres\utility

      ```
      echo %PATH%
      ```

      If you are having problems shutting down a client installation because Ingres believes a Communications Server is running locally when there is none, remove this file:

      ```
      %II_SYSTEM%\ingres\files\name\clientname\IICOMSVR_clientname
      ```

2. Identify whether Ingres is recovering aborted transactions. Details of recovery delays are described in Recovery Process Monitoring (see page 150). Issue the following command at the operating system prompt and examine the output for the word "RECOVER."

   ```
   logstat | findstr RECOVER
   ```

   If Ingres is recovering aborted transactions, wait for this process to finish. Continue reissuing the logstat command and examining the STATUS field. When it reads "ONLINE, ECP DONE," proceed with normal shutdown.

3. Shut down the installation by typing the following command:

   ```
   ingstop
   ```

   You can also click the Stop toolbar button in Ingres Visual Manager to shut down the installation.

   Note: All users must be logged out of Ingres (that is, no sessions running in the server) for ingstop to succeed.

4. Check that all processes are shut down. If shutdown succeeds, none of the following processes are running on your machine:

   - iigcn

   - iigcc

   - II_IUSV_*nnn*

   - dmfacp

   - iidbms

   - iijdbc

   - iigcd

   - iistar

   - iigcb

   - rmcmd

5. Use one of your operating system utilities to kill any of these processes that are still running.

6. Remove any shared memory segments that remain allocated to this installation. Execute the following command at the operating system prompt:

   ```
   ipclean
   ```

   Your installation is now shut down.

Check Shutdown Problems on UNIX

If Ingres has problems during shutdown on UNIX, follow these steps:

1. Check environment variables in the local user environment by entering the following commands at the operating system prompt.

    a. Verify that II_SYSTEM is set correctly:

        ```
        echo $II_SYSTEM
        ```

        /usr/r6 (this varies system by system)

    b. Check that you have the full search path to $II_SYSTEM/ingres/bin and $II_SYSTEM/ingres/utility

        ```
        echo $PATH
        ```

2. If there is more than one Ingres installation on this machine, identify your installation code by typing the following command at the operating system prompt:

    ```
    ingprenv | grep II_INSTALLATION
    ```

    The two-letter installation code is displayed (for example, the code here is JB):

    ```
    II_INSTALLATION=JB
    ```

    Take note of your installation code: _____.

3. Identify if you are shutting down a client installation or a full installation by issuing the following command at the operating system prompt:

    ```
    ingprenv | grep II_CLIENT
    ```

    If this displays "II_CLIENT=true", this is a client installation, and only two processes (iigcn, iigcc) are running on this node.

4. If you are having trouble shutting down a client installation because Ingres believes a Communications server is running locally when there is none, remove this file:

    ```
    $II_SYSTEM/ingres/files/name/clientname/IICOMSVR_clientname
    ```

    For troubleshooting details see the *Connectivity Guide*.

5. Identify whether Ingres is recovering aborted transactions. Details on recovery delays are described in Recovery Process Monitoring (see page 150). Issue the following command at the operating system prompt and examine the output for the word "RECOVER":

    ```
    logstat | grep RECOVER
    ```

    If Ingres is recovering aborted transactions, wait for this process to finish. Continue reissuing the logstat command and examining the STATUS field. When it says: "ONLINE, ECP DONE," proceed with normal shutdown.

6. Now shut down the installation:

   `ingstop`

   Note: All users must be logged out of Ingres (that is, no sessions running in the server) for the ingstop script to succeed.

7. Check that all processes are shut down:

   a. Display running processes by issuing the following command at the operating system prompt:

      BSD:

      `ps -aux | grep ingres`

      System V:

      `ps -ef | grep ingres`

   b. If shutdown succeeds, none of the following processes are running:

      - iigcn
      - iigcc
      - II_IUSV_*nnn*
      - dmfacp
      - iidbms
      - iijdbc
      - iigcd
      - iistar
      - rmcmd
      - iigcb

8. If any of these installation processes are not shut down, note the process ID of the running processes and do the following:

   a. Shut them down manually with the operating system command:

      `kill- QUIT `*`process_id`*

      where:

      process_id refers to the process ID of the process to stop.

   b. If processes are still not shut down, issue the operating system command:

      `kill -9 process_id`

      Important! *If your site has more than one Ingres installation, examine the installation code associated with the iidbms process to make sure you are stopping only the processes associated with the installation you need to shut down.*

9. Check that no shared memory segments remain allocated to this installation. Execute the operating system command:

    ```
    csreport
    ```

    The following message indicates that shared segments have been properly removed:

    ```
    !Can't map system segment
    ```

10. If shared memory segments remain for this installation, interactively remove them:

    a. To deallocate shared memory resources, issue the operating system command:

        ```
        ipcclean
        ```

    b. Use the UNIX command ipcs to verify that the actual segments have been removed:

        ```
        ipcs
        ```

    c. If they have not been properly removed, you must delete them manually.

        Important! *If your site has more than one Ingres installation, you must take care to only remove shared memory or semaphores for the installation to be shut down. If your machine contains more than one installation, enter the following command from the environment of the installation you need to target*:

        ```
        csreport
        ```

        The **csreport** utility displays the shared memory and semaphore segment identifiers for this installation.

        To remove the targeted segment(s), use the UNIX command:

        ```
        ipcrm -mmid
        ```

        or

        ipcrm -s*sid*

        where *mid* is the shared memory segment identifier and *sid* is the semaphore identifier.

11. Verify that the following files are *not* present. If they are still present, you must remove them:

    ```
    $II_SYSTEM/ingres/files/memory/lockseg.mem
    $II_SYSTEM/ingres/files/memory/sysseg.mem
    ```

    Your installation is now shut down. Instructions on how to check and restart Ingres are described in Check Ingres Installation on UNIX (see page 104).

Check Shutdown Problems on VMS

If Ingres has problems during shutdown on VMS, follow these steps:

1. Check that you are logged in as the system administrator or another privileged account by issuing the following command at the operating system prompt:

   ```
   show process
   ```

   Make sure you have the WORLD and CMKRNL privileges required for stopping Ingres processes by issuing the following command at the operating system prompt:

   ```
   show process/privilege
   ```

   A privileged account must not only have the aforementioned VMS privileges but also must have the SERVER_CONTROL privilege authorized in II_CONFIG:CONFIG.DAT.

2. Check that the logical II_SYSTEM is defined by issuing the following command at the operating system prompt:

   ```
   show logical II_SYSTEM
   ```

3. Check to see that users are logged out of Ingres. You can use the utility logstat to determine which databases are open and active and find out whom to notify of the impending shutdown. For additional information, see the Logstat section in the *Command Reference Guide*.

   To obtain a list of running processes, use the DCL command:

   ```
   show system
   ```

4. This command gives you a complete list of all processes that are running on your machine. It displays the process ID ('Pid'), Process Name, and other information. The main processes have the names as shown in Detect Ingres Startup Problems on VMS (see page 111).

   If you use the form show system/full, you can also see the owner of the process. This is helpful if you have group level installations.

5. Identify your installation code if there is more than one Ingres installation on this machine.

   For a group level installation the two-lettered code is displayed after the process name. In this example, the code is BE:

   ```
   Pid        Process Name     State Pri I/O    CPU          Page flts Ph.Mem

   236012AD II_DBMS_BE_1E3F HIB   6    86463 0 00:13:53.24 14401     8192
   ```

   Important! *If your site has more than one Ingres installation, examine the installation code associated with the II_DBMS_nnn process to make sure you are only shutting down processes associated with the installation you need to shut down.*

6. Use the iinamu utility to find the names of the servers for this installation; iinamu shows you only the servers for the installation in which you currently reside. For example:

   ```
   IINAMU> show ingres

   INGRES  *        II_DBMS_BE_1E3F

   found.

   IINAMU> show comsvr

   COMSVR  *        II_GCC_BE_2B77
   ```

7. Shut down servers. See the individual descriptions below.

   Note: Ingres can be shut down in several ways. Following are some tips on how each shutdown procedure works to enable you to make a choice as to how to shut down the installation.

   - The ingstop utility shuts down servers in the following order:
     – Remote Command Server (rmcmd)
     – Communications Server
     – Data Access Server
     – Protocol Bridge Server
     – Star Server
     – DBMS Server
     – Recovery and archiver processes-IUSV and ACP by rcp.config.
     – If present, the Cluster Server is also shut down.
     – Name Server
   - The following order also works. This ordering manually shuts down the installation, shutting down the Name Server first:
     – Name Server
     – Communications Server
     – Data Access Server

  – Star Server

  – DBMS Server

  – Recovery and archiver processes-IUSV and ACP

a. Shutting down the Name Server

When you issue the stop command from iinamu, the Name Server process is stopped. Once this command is issued, no new servers are allowed to register with the Name Server. Therefore, it makes sense to take the Name Server down first for a clean shutdown. Current, active sessions are not affected by this command.

b. Shutting down the DBMS Server

Use iimonitor to stop the DBMS Server. The two commands you can use are:

```
set server shut
```

This causes the server to refuse further connections and shuts the server down gracefully, completing all current transactions first, including the iimonitor session.

```
stop server
```

This stops the server immediately without waiting for current connected sessions to complete. Use this command instead of the VMS command stop/id.

c. Shutting down the recovery and archiver processes

Take down the recovery and the archiver processes, either quickly or slowly, depending on the method you used to take down the DBMS Server in step b. Use either of the following commands at the operating system prompt:

```
rcpconfig/shutdown
```

This causes the processes to refuse all further connections and transaction processing, but allows current transactions to finish. This allows the archiver and recovery processes to shut down cleanly.

```
rcpconfig/imm_shutdown
```

This stops all pending transactions and immediately shuts down the recovery and archiver processes. This leaves open transactions in the log file, which requires rollback (and slow startup) when the system is restarted.

d. Writing a DCL command file to take Ingres down

Alternatively, you can write a DCL script that puts the above steps in a command procedure to shut down each of the Ingres processes.

8. After shutting down Ingres, check that all processes are stopped:

a. Issue the following command at the operating system prompt:

```
show system
```

Check to be sure that none of the processes listed in the Ingstart on VMS section is shown.

b. If any of these installation processes are not shut down, note the process ID (PID#) of the running process and issue either of the following commands at the operating system prompt:

```
stop process_name
```

or

```
stop/id = pid#
```

Note: For serious system problems that require immediate attention, the VMS stop command can be used to stop an Ingres process. This is not recommended because it does not permit the recovery process to back out open transactions or the archiver process to flush the log file to disk.

If this is a cluster installation, remember that these Ingres processes run on each node.

Your Ingres installation is now shut down. Information about checking and restarting Ingres is described in Check Ingres Installation on VMS (see page 108).

Ingres Processes on Windows

The major Ingres processes are as follows:

- Name Server process (iigcn)
- Communications Server process (iigcc)
- Recovery process (II_IUSV_*nnn*)
- Archiver process (dmfacp)
- DBMS Server process (iidbms)
- Data Access Server process (iigcd)
- Bridge Server process (iigcb)
- Remote Command process (rmcmd)

Check for Name Server Errors on Windows

The Name Server process (iigcn) is not running if either of the following occurs:

- You receive a specific error indicating that the Name Server failed to start.

- The iigcn.exe does not appear in the process list.

You can verify this by attempting to start the Name Server manually.

The most common causes of Name Server startup failure are:

- An improper installation environment exists. Be sure to verify the installation environment by performing the Check Ingres Installation on Windows (see page 102) before attempting to start the Name Server.

- The Name Server fails to start if there is already a Name Server running for this installation.

If you still have problems, set the following trace to capture additional diagnostic data before calling technical support:

```
ingsetenv II_GC_TRACE 4
ingsetenv II_GC_LOG filename
iirun iigcn
```

Check for Communications Server Process Errors on Windows

If the Communications Server process (iigcc) does not start, verify that you have installed the Ingres Net component and that you have installed the correct protocol drivers.

If you are still having problems, set the following trace to capture additional diagnostic data before calling technical support:

```
ingsetenv II_GC_TRACE 4
ingsetenv II_GC_LOG filename
iirun iigcc
```

Check for Bridge Server Errors on Windows

If the Bridge Server process (iigcb) does not start, verify that you have installed the Ingres Net component and that you have installed the correct protocol drivers.

If you are still having problems, set the following trace to capture additional diagnostic data before calling technical support:

```
ingsetenv II_GC_TRACE 4
ingsetenv II_GC_LOG filename
ingstart -iigcb
```

Check for Remote Command Process Errors on Windows

You can check for the presence of the Remote Command process (rmcmd) using the Windows Task Manager. View the Processes tab for the existence of the process.

Check for Recovery Process Errors on Windows

The Recovery process (II_IUSV_*nnn*) must be running before a DBMS Server can be started. Failure of the II_IUSV_*nnn* starting process indicates one of the following:

- Improper installation configuration, as described in Problems with Startup, Shutdown, or Configuration.

- Problems with the log file

- Insufficient shared memory

If the Recovery process does not start, perform the following procedure:

1. Check for the existence of the transaction log by opening the Primary Transaction Log window in Configuration Manager (or the Transaction Log screen in the Configuration-By-Forms (cbf) utility), and noting the directories listed in the Log File Root Locations table in Configuration Manager (or the Primary Transaction Log Locations table in cbf).

2. In the ingres/log directory (located below all other listed directories), look for the file ingres_log.l*nn* where *nn* is an integer between 1 and 16.

3. Make the following checks on the transaction log file ingres_log.l*nn*:

    a. Verify that ingres_log.l*nn* exists at that location by entering the following command at the operating system prompt:

        ```
        dir
        ```

    b. If it does not exist, you must create it using the iimklog command, and format it using the rcpconfig -init command.

4. If you corrected a recovery process problem, verify that Ingres starts normally by completing the following steps:

    a. Shut down the partially started installation with the ingstop command.

    b. Restart the installation with the ingstart command.

    If you still cannot start the II_IUSV_*nnn*, you must erase it and recreate and reformat it. This step is done only as a last resort.

    Note: This step reinitializes the log file, causing any outstanding transactions to be lost.

Check for Archiver Process Errors on Windows

The archiver process (dmfacp) does not start unless the II_IUSV_*nnn* recovery process is running. However, an Ingres installation runs without an archiver process until the log file fills up. The user programs are suspended until any outstanding transactions in the log file are backed out. More information about Ingres in a recovery state is provided in Recovery Process Monitoring (see page 150).

Archiver process (dmfacp) startup errors are likely to result from:

- Improperly shared memory resource

- Inability to read the transaction log file

- Inability to write journal files

Use the following procedure to check archiver process startup problems. Some of these checks are the same as for the recovery process in the previous section.

1. Make the following check on the transaction log file ingres.log:

    Verify that ingres.log exists at that location by entering the following command at the operating system prompt:

    ```
    dir
    ```

    If it does not exist, you must create it using the iimklog command, and format it using the rcpconfig –init command.

2. Check journal locations by verifying that:

    The journal location name points to a valid directory containing subdirectories ingres\jnl\default*dbname*.

3. Check that the disk partition containing the journal files is not full. Issue the following command from the operating system prompt:

    ```
    dir
    ```

    If the journal partition is 100% full, this makes it impossible to write journals and stalls the archiver. If this is the reason preventing archiver start up, you must either free space on the journal partition or temporarily disable journaling with the alterdb command. For details on alterdb, see the *Command Reference Guide*.

4. If you corrected a recovery process problem, verify that Ingres starts normally.

    a. First, shut down the partially started installation with the ingstop command.

    b. Restart the installation with the ingstart command.

Check for DBMS Server Process Errors on Windows

The command to start the DBMS Server process (iidbms) from the Ingres login is ingstart **-**iidbms. If the DBMS Server did not start:

1. Verify that the recovery process II_IUSV_*nnn* is running, as described in Check for Recovery Process Errors on Windows (see page 125).

2. Verify that the recovery process II_IUSV_*nnn* is not in a recovery state, as described in Recovery Process Monitoring (see page 150). This is a likely situation if there was a sudden shutdown because of a power failure or some other system failure.

3. Try to start the DBMS Server. Issue the following command at the operating system prompt:

   ```
   ingstart -iidbms
   ```

4. If you corrected a recovery process problem, verify that Ingres starts normally.

   a. First, shut down the partially started installation with the ingstop command.

   b. Restart the installation with the ingstart command.

Check for Data Access Server Errors on Windows

If the Data Access Server process (iigcd) does not start, follow this procedure:

1. Verify that you have installed the Ingres Net component

2. Verify that you have installed the correct protocol drivers

If you are still having problems, set the following trace to capture additional diagnostic data before calling technical support:

```
ingsetenv II_GCD_TRACE
ingsetenv II_GCD_LOG filename
ingstart -iigcd
```

Ingres Processes on UNIX

The major Ingres processes are as follows:

- Name Server process (iigcn)
- Communications Server process (iigcc)
- Recovery process (II_IUSV_*nnn*)
- Archiver process (dmfacp)
- DBMS Server process (iidbms)
- Data Access Server process (iigcd)
- Bridge Server process (iigcb)
- Remote Command process (rmcmd)

Name Server Errors on UNIX

The Name Server process (iigcn) is not running if either of the following occurs:

- You receive a specific error indicating that the Name Server process (iigcn) failed to start.
- The command ps -aux (BSD) or ps -ef (System V) shows that the iigcn is not running.

You can verify this by attempting to start the Name Server manually.

Check for Name Server Errors on UNIX

If the Name Server does not start, follow these steps:

1. Verify that TCP/IP is properly installed by typing the following command at the operating system prompt:

   ```
   telnet localhost
   ```

   A loopback login to your machine occurs.

2. Verify that the required TCP daemon process for your operating system is running.

   The specific process name is system dependent, but on many UNIX systems, the process is named "inetd" (use your process name in the command below if it is not inetd). Issue the following command at the operating system prompt, or see your operating system manual for your particular TCP/IP implementation:

   BSD:

   ```
   ps -aux | grop inetd
   ```

   System V:

   ```
   ps -ef | grep inetd
   ```

3. Check that the Ingres environment variable II_GCNxx_PORT is not set (this environment variable contains the TCP port identifier of the Name Server process):

   a. Use the ingprenv utility to verify that this environment variable is *not* set when the Name Server tries to start up.

   b. If necessary, use the ingunset command to *unset* the II_GCNxx_PORT environment variable.

4. If you corrected a Name Server problem, verify that Ingres starts normally:

   a. Shut down the partially started installation with the ingstop command.

   b. Restart the installation with the ingstart command.

5. If you are still having problems, set the following trace to capture additional diagnostic data before calling technical support:

   Bourne Shell:

   ```
   II_GC_TRACE=5
   II_GC_LOG = stdio (stdio or filename); export II_GC_TRACE II_GC_LOG
   ```

   C Shell:

   ```
   setenv II_GC_TRACE 5
   ```

   ```
   setenv II_GC_LOG stdio       (stdio or filename)
   ```

   ```
   iirun iigcn
   ```

Check for Communications Server Process Errors on UNIX

If the Communications Server process (iigcc) did not start, follow this procedure:

1. Verify that no local environment variables are set in the local user environment that contain the string "GCC". At the operating system prompt type the command:

   BSD:

   ```
   printent | grep GCC
   ```

   System V:

   ```
   env | grep GCC
   ```

   The Ingres environment variable II_GCNxx_PORT is set in the Ingres symbol table. It is *not* visible from the UNIX environment using the printenv or env commands, but is visible in the Ingres environment. Verify this by typing **ingprenv.**

2. Check the value of the Ingres environment variable II_RUN with the following command entered at the operating system prompt:

   ```
   ingprenv
   ```

   a. If this machine is running Ingres Net, the value of II_RUN is either ",NET" or ",DBMS, NET". If the value of the Ingres environment variable II_CLIENT is TRUE, II_RUN is ",NET". If II_RUN is not set correctly, reset it using the command:

   ```
   ingsetenv II_RUN ', DBMS, NET'
   ```

   b. If this machine is an NFS client (that is, does not run a DBMS Server locally) the value of II_RUN is ",NET". If II_RUN is not set correctly, reset it using the following command entered at the operating system prompt:

   ```
   ingsetenv II_RUN', NET'
   ```

   For details on setting environment variables, see the chapter "Setting Environment Variables and Logicals."

3. If you are still having problems, set the following trace to capture diagnostic data and attempt to restart the Communications Server:

   Bourne Shell:

   ```
   II_GCC_TRACE=4
   II_GCA_LOG=stdio
   export II_GCC_TRACE II_GCA_LOG
   iirun iigcc
   ```

C Shell:

```
setenv II_GCC_TRACE 4

setenv II_GCA_LOG stdio

iirun iigcc
```

4. If you corrected a Communications Server problem, verify that Ingres starts normally:

    a. Shut down the partially started installation with the ingstop command.

    b. Restart the installation with the ingstart command.

Check for Bridge Server Process Errors on UNIX

If the Bridge Server process (iigcb) did not start, verify that you have installed the Ingres Net component and that you have installed the correct protocol drivers.

If you are still having problems, set the following trace to capture additional diagnostic data before calling technical support:

```
ingsetenv II_GC_TRACE 4
ingsetenv II_GC_LOG filename
ingstart -iigcb
```

Recovery Process Errors on UNIX

The recovery process (dmfrcp) must be running before a DBMS server can be started. Failure of the dmfrcp starting process indicates one of the following:

- An improper installation configuration exists, as described in Troubleshoot Startup, Shutdown, or Configuration Problems (see page 100).

- Problems with the log file

- Insufficient or previously allocated system semaphores or shared memory

- If configured for Ingres Cluster Solution, check that required OS prerequisites are installed and operational (for example, OpenDLM in a Linux environment).

Check for Recovery Process Errors on UNIX

If the recovery process does not start, perform the following procedure:

1. Check that the shared memory resources are properly installed. Use the csreport utility and check that the size, ownership, and permissions of the semaphore and shared memory segments meet the minimum requirements for your port.

2. Check for the existence of the transaction log by opening the Primary Transaction Log window in Configuration Manager (or the Transaction Log screen in the Configuration-By-Forms (cbf) utility), and noting the directories listed in the Log File Root Locations table in Configuration Manager (or the Primary Transaction Log Locations table in cbf).

3. Look for the file ingres_log.l*nn* (where nn is an integer between 1 and 16) in the ingres/log directory (located below all other listed directories).

4. Make the following checks on the transaction log file ingres_log.l*nn*:

   a. Verify that ingres_log.l*nn* exists at that location by entering the following command at the operating system prompt:

      ```
      ls -l
      ```

      If it does not exist, you must recreate it using the Configuration-By-Forms (cbf) or Configuration Manager (vcbf) utility.

   b. Verify that ingres_log.l*nn* is owned by installation owner.

      If it is not, issue the following command at the operating system prompt, where *userid* is the user ID of the installation owner:

      ```
      chown userid ingres_log
      ```

   c. If the transaction log file was created as:

      An ordinary UNIX file, make sure it has permissions 660 (that is, "-rw-rw----"). If not, issue the following command at the operating system prompt:

      ```
      chmod 660 ingres_log
      ```

      A raw log, permissions is "crw------".

5. Verify that Ingres starts normally.

   a. Shut down the partially started installation with the ingstop command.

   b. Restart the installation with the ingstart command.

6. If you still cannot start the II_IUSV_*nnn* process, you need to completely shut down the Ingres installation, re-run ingbuild, and reconfigure the log file.

   Important! *This step must only be done as a last resort. Keep in mind that this reinitializes the log file, and any outstanding transactions are lost.*

To shut down the system, complete the following steps:

a. Issue the command **ingstop**.

b. Check Shutdown Problems on UNIX (see page 116). Work through that section until your Ingres installation has been cleanly shut down.

Check for Remote Command Process on UNIX

You can check for the presence of the remote command process by entering the following command at the operating system prompt:

```
ps -ef | grep rmcmd
```

Archiver Process on UNIX

The archiver process (dmfacp) does not start unless the recovery (dmfrcp) process is running. However, an Ingres installation runs without an archiver process until the log file fills up. User programs are suspended as outstanding transactions in the log file are backed out. For information about the Ingres recovery state, see Recovery Process Monitoring (see page 150).

- Archiver process (dmfacp) startup errors are likely to result from:

- Improper shared memory resources

- Inability to read the transaction log file

- Inability to write journal files

Check for Archiver Process Errors on UNIX

Use the following procedure to check archiver process startup problems. Some of these checks are the same as for the recovery process:

1. Check that the shared memory resources are properly installed: Use the csreport utility and check that the size, ownership and permissions of the semaphore and shared memory segments meet the minimum requirements for your port. For requirements, see the Readme file.

   The command to allocate Ingres shared memory and semaphores (when logged is as the installation owner) is csinstall. You can display them from UNIX with the command ipcs.

2. Make the following checks on the Ingres transaction log file "ingres_log":

   a. Verify that "ingres_log" exists at that location by entering the following command at the operating system prompt:

      ```
      ls -l
      ```

      If it does not exist, you must create it by running the ingbuild program.

   b. Verify that "ingres_log" is owned by the user that owns the installation.

      If not, issue the following command at the operating system prompt, where *userid* is the user who owns the installation:

      ```
      chown userid ingres_log
      ```

   c. If the transaction log file was created as:

      An ordinary UNIX file, make sure it has permissions 660 (that is, "-rw-rw----"). If not, issue the following command at the operating system prompt:

      ```
      chmod 660 ingres_log
      ```

      A raw log, permissions is "crw------".

      Check that the II_JOURNAL location is a valid location. Issue the following command at the operating system prompt:

      ```
      infodb | grep ii_journal
      ```

3. Check journal locations by verifying that:

   a. The journal location name points to a valid directory containing subdirectories "ingres/jnl/default/dbname"

   b. The permissions on these directories are:

      - 755 for the ingres directory
      - 777 for the default directory

4. Check that the disk partition containing the journal files is not 100% full. Issue the following command at the operating system prompt:

```
df
```

If a journal partition is 100% full, it is impossible to write journals and the Archiver stalls. If this is the reason preventing Archiver startup, you must either free space on the journal partition or temporarily disable journaling with the alterdb command. For details on alterdb, see the *Command Reference Guide*.

5. If you corrected an Archiver process problem, verify that Ingres starts normally:

   a. Shut down the partially started installation with the ingstop command.

   b. Restart the installation with the ingstart command.

Check for DBMS Server Process Errors on UNIX

The command to start the DBMS Server process (iidbms) from the installation owner login is ingstart -iidbms. If the DBMS Server did not start:

1. Verify that the recovery process (dmfrcp) is running. Details are descxribed in Recovery Process Errors on UNIX (see page 131).

2. Verify that the recovery process is not in a recovery state (see page 150). This is a likely situation if there was a sudden shutdown because of a power failure or some other systems failure.

3. Try to start up a DBMS server. Issue the following command at the operating system prompt:

```
ingstart -iidbms
```

4. If you corrected a DBMS server problem, verify that Ingres starts normally:

   a. Shut down the partially started installation with the ingstop command.

5. Restart the installation with the ingstart command.

Check for Data Access Server Process on UNIX

The command to start the Data Access Server process (iigcd) from the installation owner login is ingstart -iigcd. If the Data Access Server did not start:

1. Verify that the recovery process (dmfrcp) is running. More information is provided in Check for Recovery Process Errors on UNIX (see page 132).

2. Verify that the recovery process (see page 150) not in a recovery state. This is a likely situation if there was a sudden shutdown because of a power failure or some other systems failure.

3. Try to start up a Data Access server. Issue the following command at the operating system prompt:

   ```
   ingstart -iigcd
   ```

4. If you corrected a Data Access server problem, verify that Ingres starts normally:

   a. Shut down the partially started installation with the ingstop command.

   b. Restart the installation with the ingstart command.

Ingres Processes on VMS

The major Ingres processes are as follows:

- Name Server process (iigcn)

- Communications Server process (iigcc)

- Recovery process (II_IUSV_*nnn*)

- Archiver process (dmfacp)

- DBMS Server process (iidbms)

- Data Access Server process (iigcd)

- Bridge Server process (iigcb)

Remote Command process (rmcmd)

Name Server Errors on VMS

The Name Server (II_GCN) is not running if the following occurs:

- You receive a specific error indicating that the Name Server process failed to start.

- The VMS command show system does not list the II_GCN process, indicating that the Name Server is not running.

    Note: Only a privileged user can start up the Name Server.

Check for Name Server Errors on VMS

If the Name Server does not start, follow these steps:

1. Examine the II_CONFIG:errlog.log file for any additional information.

2. To isolate a running Name Server as the problem on failed local connections:

    a. Define the logical II_DBMS_SERVER to point to the DBMS Server process name or mailbox.

    b. If the connection now succeeds, stop and start the Name Server. For additional information about the iinamu command, see the *Command Reference Guide*.

3. If the Name Server continues to fail, contact technical support, as described in What You Need Before Contacting Technical Support (see page 170).

Communications Server Process on VMS

The Communications Server (II_GCC_*nnn*, where the *nnn* is a number related to the server's process ID) does not start if any of the following errors occur:

- The II_CONFIG:errlog.log file has a notation regarding the failure of the "net server" (Communications Server) process starting.

- The show system command does not list an II_GCC_*nnn* process.

- The utility iinamu does not display a process name when you enter the show comsvr command at the IINAMU> prompt.

Check for Communications Server Process Errors on VMS

If the Communications Server does not start, follow these steps:

1. Examine the II_CONFIG:errlog.log file for any information regarding Communications Server startup.

2. Make sure that the network protocol software on which the Communications Server executes is started and functioning before attempting to start Ingres Net processes. By default, an attempt is made to open ports for the DECnet and TCP_DEC protocols.

3. If the Communications Server continues to fail to start, contact Technical Support.

Bridge Server Errors on VMS

The Bridge Server process (II_GCB_*nnn*, where the *nnn* is a number related to the server's process ID) does not start if the following errors occur:

- The II_CONFIG:errlog.log file has a notation regarding the failure of the Bridge Server process.

- The show system command does not list a II_GCB_*nnn* process.

- The utility iinamu does not display a process name when you enter the show bridge command at the IINAMU> prompt.

Check for Bridge Server Process Errors on VMS

If Bridge Server does not start, follow these steps:

1. Verify that you have installed the Ingres Net component.

2. Verify that you have installed the correct protocol drivers and they are functioning properly.

3. Verify the Bridge Server has been properly configured using the Configuration-By-Forms (cbf) utility. If you are still having problems, set the following trace to capture additional diagnostic data before calling technical support:

```
define II_GC_TRACE 4
define II_GC_LOG filename
ingstart -iigcb
```

Recovery Process Errors on VMS

The recovery process (II_IUSVxx_*nnn* for stand-alone Ingres, or DMFRCPnnn-xx for Ingres Cluster Solution) must be running before a DBMS server can be started. Failure of the recovery process to start indicates one of the following:

- Improper installation configuration. Check Ingres Installation on VMS (see page 108).

- Problems with the log file

- Insufficient system memory (NPAGEDYN sysgen parameter)

Check for Recovery Process Errors on VMS

If the recovery process does not start, perform the following procedure.

1. Examine the following files for specific reasons for the startup failure:

   II_SYSTEM:[INGRES]II_CONFIG:errlog.log
   II_CONFIG:IIRCP.LOG

   (If your system is configured for the Ingres Cluster Solution, the suffix *_nodename* is added.)

2. If configured for Ingres Cluster Solution, check that DECnet is up among all nodes. Ingres uses DECnet to communicate among nodes. Check that you can set host and log into/from all nodes running Ingres.

3. Be sure that the log file location is correctly defined:

   a. Check the definition of the II_LOG_FILE logical.

   b. Check the II_LOG_FILE location for the existence of the log file:

   II_LOG_FILE:[INGRES.LOG]INGRES_LOG.SYSTEM_*nodename*

4. If you still cannot start the recovery process, the problem may be a corrupted log file. Contact technical support, as described in What You Need Before Contacting Technical Support (see page 170).

Remote Command Process Errors on VMS

The Remote Command Server (RMCMD_nnn, where the nnn is a number related to the server's process id) does not start, if show system command does not list an RMCMD_*nnn* process.

Check for Remote Command Server on VMS

Use the following procedure if the RMCMD process did not start:

1. Ensure that the PRCLM for the system administrator is sufficient to start this additional process. This value must be a minimum of 15.

2. Check II_SYSTEM:[ingres.bin]RMCMD.log for errors.

Archiver Process Errors on VMS

The Archiver process (DMFACP or DCMFACPxx) does not start if any of the following conditions occur:

- Unless the recovery process is running; however, an installation runs without an archiver process until the log file fills up. User programs are suspended until the archiver is started.

- When the recovery system is backing out aborted transactions from a system failure. Recovery Process Monitoring (see page 150) can help you determined whether Ingres is in a recovery state.

- Due to improper memory resources, inability to read the transaction log file, or inability to write journal files.

- You receive an error indicating that the Archiver process failed to start.

- The VMS command show system does not list the Archiver process.

Check for Archiver Process Errors on VMS

Use the following procedure to check Archiver process startup problems:

1. Examine the II_CONFIG:errlog.log startup log file for information.

2. Examine the IIACP.LOG_*nodename* file for possibly specific reasons for the startup failure. Take note of which databases are referenced in the IIACP.LOG_*nodename* file.

3. Check that the II_JOURNAL location is a valid location:

    a. Check the definition of the II_JOURNAL logical.

    b. Check the II_JOURNAL location. Type the following command at the operating system prompt:

        ```
        directory II_JOURNAL:[INGRES.JNL]
        ```

4. Check that the disk containing the journal files is not 100% full. This makes it impossible to write journals and stops the Archiver.

    If the journal disk became full as a result of the Archiver writing journal information, an error appears in the errlog.log stating a "device full" condition.

    a. After making disk space available, restart the Archiver using the ingstart -dmfacp command.

    b. Optionally, modify the acpexit script to take whatever measures are necessary to notify you of this situation as soon as it occurs. For example, you can modify the script so that it sends you mail when the Archiver encounters an error.

5. If the journal files have been corrupted, the errlog.log reports any problems with the Archiver writing information to the journal files.

If there are corrupted files, or the Archiver process continues to fail, contact technical support, as described in What You Need Before Contacting Technical Support (see page 170).

Check for DBMS Server Process Errors on VMS

If the DBMS server (II_DBMS_*nnn*, where the *nnn* is a number related to the server's process ID) does not start, follow these steps:

1. Examine the II_CONFIG:errlog.log file for any information regarding DBMS Server startup.

    Contact Technical Support to examine your authorization string.

2. If there is a DBMS Server startup failure reported in the II_CONFIG:errlog.log, but no error reported in the errlog.log:

    a. Check for the existence of any MBA*nnn* processes on the system that are linked to the IIRUNDBMS.EXE*.* Examine the MBA*nnn* process by issuing the following command at the operating system prompt:

        ```
        show proc/id = pid /cont
        ```

        where*:*

        pid is the process ID of the MBA*nnn* process.

    b. If the MBA*nnn* process is linked to the IIRUNDBMS.EXE, free the IIRUNDBMS.EXE by issuing the following command at the operating system prompt:

        ```
        stop /id = pid
        ```

        where:

        pid is the process ID of the MBA*nnn* process.

    c. Check the errlog.log for any new messages relating to DBMS Server startup failure.

3. Try starting a server manually, using the ingstart -iidbms command.

4. If the DBMS Server continues to fail to start, contact Technical Support.

Data Access Server Errors on VMS

The Data Access Server (II_GCD_*nnn*, where the *nnn* is a number related to the server's process ID) does not start if any of the following conditions occur:

-   The II_CONFIG:errlog.log file has a notation regarding the failure of the Data Access Server process.

-   The command **show system** does not list a II_GCD_*nnn* process.

-   The utility iinamu does not display a process name when you enter the command show dasrv at the IINAMU> prompt.

Check for Data Access Server Errors on VMS

Use the following procedure if the Data Access Server process did not start:

1. Verify that you have installed the Ingres Net component.

2. Verify that you have installed the correct protocol drivers and they are functioning properly.

3. Verify the Data Access Server has been properly configured through the Configuration-By-Forms (cbf) utility.

4. If you are still having problems, set the following trace to capture additional diagnostic data before calling technical support:

```
define II_GCD_TRACE 4
define II_GCD_LOG filename
ingstart -iigcd
```

Problems with Tools Startup

The following flow chart helps you isolate a problem when starting an Ingres tool:
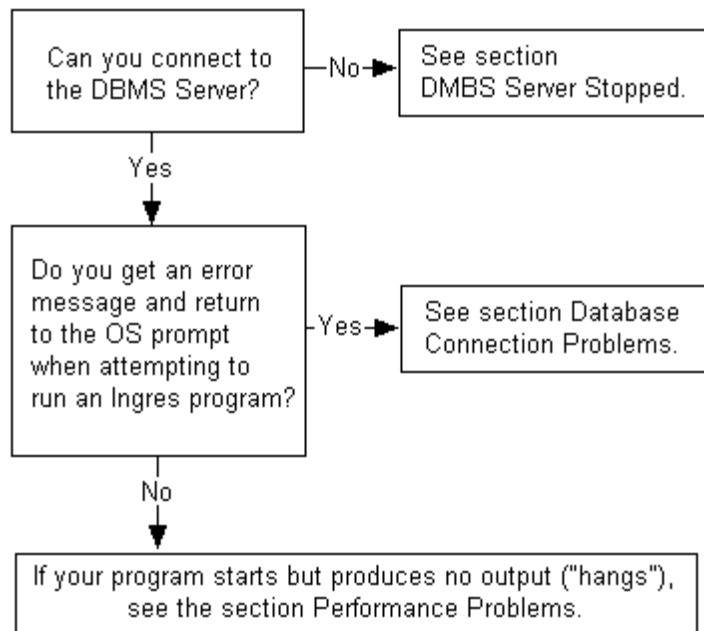
DBMS Server Stopped

Once started with ingstart, the DBMS Server process must continue running until the ingstop or iimonitor command is issued to stop it. If the DBMS Server stops running ("dies") for any other reason, report it to technical support along with the associated error log messages and, if possible, the cause of the DBMS Server stopping.

1. Document error log entries associated with the process death. Details on reading the log files are described in Check the Error Log Files (see page 99). Save all errors for technical support.

2. Isolate the reason your DBMS Server process died.

   a. Isolate which operations, application, query, and tables are needed to duplicate the problems. See the copyapp and unloaddb command descriptions in the *Command Reference Guide*.

   b. Save this to make a test case for technical support.

3. If the immediate cause cannot be isolated, perform long-term diagnostics with II_DBMS_LOG. This diagnostic tool is especially valuable for fatal DMF errors.

   a. Set II_DBMS_LOG to capture a "snapshot" of the DBMS Server when it stops by setting it to the full path name of a file before starting the DBMS Server. For example:

   Windows:

   ```
   SET II_DBMS_LOG=%II_SYSTEM%\ingres\files\dbms.log
   ```

   UNIX:

   C Shell:

   ```
   setenv II_DBMS_LOG $II_SYSTEM/ingres/files/dbms_%p.trace
   ```

   Bourne Shell:

   ```
   II_DBMS_LOG = $II_SYSTEM/ingres/files/dbms_%p.trace export
   ```

   ```
   II_DBMS_LOG
   ```

   At startup, the %p in the II_DBMS_LOG specification is replaced by the Process Identifier (PID) of the server process. This prevents DBMS servers from clobbering each others logs (or the recovery process log)

   VMS:

   ```
   define /system/exec II_DBMS_LOG -
   II_SYSTEM:[INGRES.FILES]DBMS.LOG  (for a system level installation)
   ```

   define/group/exec II_DBMS_LOG -
   II_SYSTEM:[INGRES.FILES]DBMS.LOG (for a group level installation)

b. When the DBMS Server shuts down, information is dumped to the DBMS log file. You must rename the file before restarting Ingres or the new server overwrites the file. Prepare to send this file along with associated errors from the error log files to technical support for analysis.

Database Connection Problems

Database connection problems occur in the following scenarios:

- If you cannot connect to any database

- If you encounter errors while connecting to individual databases

No Database Connections

If you cannot connect to *any* database, follow these steps:

1. Check the errlog.log to see if there are any associated messages. These messages are often more informative than the message displayed on your screen and can quickly identify the source of failure. Check for associate messages in the errlog.log by using their timestamps.

2. Ensure that all your installation's processes are running, as described in Check Ingres Installation on Windows (see page 102), Check Ingres Installation on UNIX (see page 104), and Check Ingres Installation on VMS (see page 108).

3. Run logstat to insure that the logging system status is ONLINE and not LOGFULL. If the status is not ONLINE, see How to Log System Issues (see page 160).

4. Use iinamu to interrogate the Name Server. (For additional information, see iinamu in the *Command Reference Guide*). Type **show ingres** to verify that the DBMS Server has registered with the Name Server.

5. Use the iimonitor utility to see if you can connect to the DBMS Server. See iimonitor in the *Command Reference Guide*.

    If you *can* connect, at the "IIMONITOR>" prompt type **show sessions** to examine DBMS server activity.

    If you *cannot* connect to the DBMS Server, see How to Log System Issues (see page 160).

    Check the Ingres environment variable/logical II_DBMS_SERVER:

    Windows:

    Use ingprenv and your system's echo command to verify that II_DBMS_SERVER is not set, either in the Ingres symbol table or your local environment.

    UNIX:

    Use ingprenv and your system's env or printenv command to verify that II_DBMS_SERVER is not set, either in the Ingres symbol table or your local environment.

    VMS:

    Use II_DBMS_SERVER to bypass the Name Server process. Issue the following command at the operating system prompt:

    ```
    define II_DBMS_SERVER II_DBMS_pid
    ```

6. Restart the Name Server if II_DBMS_SERVER works.

7. If the error condition persists, contact technical support, as described in What You Need Before Contacting Technical Support (see page 170).

Individual Database Connection Failure

If you can connect to *some* databases but not others:

1. Check the errlog.log for database connection error messages. These messages can quickly identify the source of the failure.

2. Check that the database is not exclusively locked by another user.

3. Check database permissions and ownership, as described in Check Ingres Installation on Windows (see page 102), Check Ingres Installation on UNIX (see page 104), and Check Ingres Installation on VMS (see page 108).

   Verify that your database still exists and is not being recovered by the recovery system.

4. Type **catalogdb** and choose "catalog" / "databases" or run VDBA to check if the database is listed.

   If the database does not exist, a corresponding error message is sent.

   Using VDBA:

5. Select a server and click Connect DOM.

6. Select Database from the drop down menu and choose infodb. Make sure the database is not listed as inconsistent and that the status indicated is "VALID."

Inconsistent Databases and Recovery

An inconsistent database occurs when administrative changes to a database in the transaction log do not agree with information maintained in the database's configuration file.

The main causes of inconsistent database errors are improper system administration procedures. These include:

- Initializing the transaction log file with the **-**force_init_log flag of rcpconfig while the log file still contains open transactions

- Moving or altering files or installation variables without using the appropriate utility such as ingbuild or unloaddb

- Improper procedures when recovering Ingres data from operating system backups

Inconsistent database errors can also be caused by hardware problems or software problems. For example, inconsistent database errors can be caused when the transaction log file or the configuration file has been corrupted by a hardware failure. Inconsistent database errors can also be caused by software bugs. In either case, contact technical support.

Note: You can also use VDBA for database backup and recovery.

Automatic Recovery

Ingres automatically handles the transaction failures that cause most database inconsistencies.

Recovery During Normal Operation

If a user program exits or a transaction is aborted for some other reason, the DBMS Server automatically handles transaction rollback. This does not cause an inconsistent database.

Recovery at Shutdown

At shutdown, all users must have exited their sessions; therefore, all transactions are committed. If users exited their sessions abnormally, the DBMS Server aborts any open transactions associated with the aborted sessions. Very long transactions take time to roll back and cause ingstop to seem to hang. The DBMS Server process cannot exit normally until it finishes recovering the aborted transactions.

If transactions are being rolled back on shutdown, allow the DBMS Server to finish this task before shutting down. If you do not, longer delays occur at startup time while the recovery process is performing rollback.

Recovery at Startup

If transactions have been aborted and were not recovered by a normal shutdown, upon restart the recovery process performs recovery. This occurs, for example, if:

- Processes are forcibly killed from the operating system

- The machine is rebooted

- Power to the system is interrupted

The recovery process performs the following steps upon startup:

1. Reads the transaction log file. If there has not been a normal shutdown, the recovery process detects that databases are inconsistent—that is, that Ingres previously exited without completing all the transactions required for system and database consistency.

2. Proceeds through the transaction log file to back out uncommitted transactions and complete committed fast-commit transactions until the databases are again in a consistent state. While recovery is proceeding, no user interfaces can connect to a database.

Recovery actions are logged in the file $II_SYSTEM/ingres/files/iircp.log.

Note: When configured for the Ingres Cluster Solution, the log name has _nodename_ appended to the base log name. When the recovery process is complete, the message "Completed aborting transactions" is written to the RCP log file. Restart Ingres with the ingstart command. Users can reconnect to the databases.

Recovery Process Monitoring

If you are monitoring Ingres startup after a machine reboot, the following messages are displayed:

```
Starting Ingres Name Server...
Starting Ingres Communications Server...
Starting Ingres Recovery Process...
```

If the transaction log contained uncommitted transactions when the machine failure occurred, the startup script pauses while the recovery process recovers transactions from the transaction log file. No messages are printed to the screen.

If you are in doubt as to whether recovery is taking place during startup, or to monitor the recovery process, use the following procedure.

Display the recovery process log file by typing the following command at the operating system prompt:

Windows:

```
%II_SYSTEM%\ingres\files\iircp.log
```

UNIX:

```
tail -f iircp.log
```

VMS:

```
II_SYSTEM:[INGRES.FILES]IIRCP.LOG_nodename
```

If the system is recovering, the recovery actions are logged to the IIRCP.LOG file. This indicates that Ingres is automatically recovering from possible inconsistencies.

Messages are printed to the log file during recovery:

- The message at the beginning of recovering transactions indicates that transaction recovery has begun.

- Intermediate messages track recovery progress. As recovery proceeds, progress messages (for example, "Recovered 31 of 130 transactions") are displayed.

- When done, the following message is printed:

  Recovery complete.

You can also use an operating system command to determine whether the recovery process is recovering transactions by checking to see if it is accumulating CPU time.

VMS:

You can issue the following command at the operating system prompt several times:

```
show process/cont/id=xxxx
```

where:

xxxx is the process ID of the recovery process.

If the CPU time and disk I/O count taken by the process is increasing, the recovery process is most likely recovering transactions from the log file.

UNIX:

On UNIX, you can also monitor the files in the database directory of the database you suspect of being the target of the updates that are being backed out. The following command entered at the operating system prompt shows whether data files are being updated:

```
ls -lt
```

The file most recently updated is listed first along with the time of last update.

If any of the monitoring techniques above indicate that transaction recovery is taking place, continue to monitor the recovery process until recovery has completed. When the recovery process is complete, CPU time is not accumulated.

After the recovery process has finished, restart the installation with the ingstart command. The ingstart utility first shuts down and brings up all required installation processes. Programs can connect to the databases.

Inconsistent Database

If you receive an "inconsistent database" error after recovery is complete, it means updates and modifications were not properly completed or rolled back, and the database is therefore in an inconsistent state.

Following are examples of "inconsistent database" errors that indicate your database has become inconsistent:

```
E_DM0100 DB_INCONSISTENT Database is inconsistent
E_US0026 Database is inconsistent. Please contact the ingres system manager
E_DM9327 BAD_OPEN_COUNT
```

Diagnose an Inconsistent Database

Diagnose the cause and extent of an inconsistent database problem before you attempt to recover your database. Knowing the cause of the problem is essential to choosing the proper recovery procedures. Once a database has been rolled forward from a checkpoint, recovered from an operating system backup, or forced consistent, you cannot determine the cause of inconsistency.

To diagnose the cause and extent of an inconsistent database problem:

1. Read and save the full text of the error messages in errlog.log and iircp.log.

2. Using VDBA, choose infodb from the Database menu to read the database's configuration file and identify the cause of inconsistency. You can also enter the infodb command at the operating system prompt.

   If the configuration file can be opened and read, the cause of the inconsistency is displayed. Save the output of infodb for technical support.

   If the database's configuration file, "aaaaaaaa.cnf", cannot be read, it is corrupted. You need to recover from a backup, as described in Recover an Inconsistent Database (see page 157).

3. Review the history of your Ingres installation. Look for improper system administration procedures that have caused the database to become inconsistent. See the table in Common Causes of Inconsistent Databases (see page 152).

4. Report your problem to technical support. If inconsistent database was not caused by incorrect system administration procedures, hardware failure, or known operating system software bugs, record the information, as discussed in What You Need Before Contacting Technical Support (see page 170).

Common Causes of Inconsistent Databases

Common causes of inconsistent databases are:

- Operating system backups
- Incorrect installation paths
- Disabling of logging/eecovery system
- Use of unsupported hardware configuration

Inconsistencies Due to Operating System Backups

To recover a database from an operating system backup that was made while the installation was running, see Recover an Inconsistent Database (see page 157).

Inconsistencies Due to Incorrect Installation Paths

Changing Ingres installation variables (such as II_SYSTEM, II_DATABASE, II_CHECKPOINT, II_JOURNAL, II_DUMP, or II_WORK) without using proper procedures, causes inconsistency between the information stored in the installation variables and those stored in the database configuration file "aaaaaaaa.cnf".

Database inconsistency can occur if you move a database, table, application or some other object by using operating system commands rather than the supported Ingres utilities. If the inconsistency is the result of moving a database from another location or installation without using unloaddb, you must remove the database using destroydb, recreate the database using createdb, and repopulate the database using the unloaddb utility. For details, see the *Database Administrator Guide.*

A database file can become corrupted from hardware or software failures of various kinds. A data file can be inadvertently deleted by hand, but this is rare because only the user who owns the installation can write to the database directories.

If you are in doubt about whether transactions are being recovered, run the logstat utility and examine the "Status" field. It is marked RECOVER if in the recovery state. While a recovery is taking place, for example when restarting after a system failure, the recovery process requires time to read through the transaction log file to back out uncommitted transactions and complete fast-commit transactions. To users, the system appears to hang.

Examine Configuration File of a Database

To examine the configuration file of your database, use the VDBA Database, Infodb menu command. You can also enter the infodb command at the command prompt.

1. Compare the path information for the checkpoint, journal, data and dump locations with that defined for these environment variables and logicals as displayed by the following command:

   Windows:

   ```
   ingprenv
   ```

   UNIX:

   ```
   ingprenv
   ```

   VMS:

   ```
   show log ii
   ```

2. Return the installation logicals to the values displayed to by infodb, if the values have chanded. If these values are not the same, the installation logicals have been changed, or the database has been imported from some other Ingres installation.

3. If you need to change the existing values of Ingres installation variables or import a database from another site, you must use the unloaddb utility, as this creates a new, up-to-date configuration file for the database. For a discussion of Ingres environment variable and logicals that cause an inconsistent database if changed after installation is completed, see the chapter Environment Variables and Logicals.

Recovery Rules

The following are some rules that you should keep in mind about the recovery of transactions:

- It takes at least as long to recover aborted transactions as it took to execute them originally.

- The amount of time required for recovery depends on the number of users and transactions, transaction semantics (whether autocommit is set), and the consistency point interval.

- While recovery is proceeding, all users are denied access to databases. Any attempt to connect to a database at this point returns an error such as the following:

  `E_LQ0001_STARTUP gca protocol service request failure.`

- Database inconsistency can occur if a user or the system administrator attempts to "force" entry into the installation by running rcpconfig with the -force_init_log flag (thus erasing the transaction log file) before the recovery system has finished rolling back the uncommitted transactions during recovery.

After a system failure, monitor recovery and always allow it to proceed until the "Recovery Complete" message appears in iircp.log.

Inconsistencies Due to Disabling of Logging or Recovery System

By disabling the logging or recovery system, the DBA can temporarily turn off logging for the database to speed bulk loading of data. If logging has been turned off for this database, a NOLOGGING error message appears in the error log file. Typically this message is:

E_DM9050_TRANSACTION_NOLOGGING Database dbname has been updated by a session running with SET NOLOGGING defined.

If the database has become inconsistent, you can check for this error message by typing the following command:

Windows:

```
findstr DM9050\ %II_SYSTEM%\ingres\files\errlog.log
```

UNIX:

```
grep DM9050 \$II_SYSTEM/ingres/files/errlog.log
```

VMS:

```
search -
II_SYSTEM:[INGRES.FILES]ERRLOG.LOG -
DM9050
```

If the NOLOGGING error message appears, logging *was* disabled on this database. If the NOLOGGING message in the error log was written later than the most recent checkpoint of this database, the database must be restored from the checkpoint. To determine if this is the case, compare the timestamp on the error message in errlog.log with the timestamp in the "checkpoint history" field of the output from the command infodb *dbname*.

For details on set nologging, see the *Database Administrator Guide*.

Database Inconsistencies Due to Use of Unsupported Hardware Configurations in UNIX

Database inconsistencies can be caused by the use of unsupported hardware configurations on NFS. In systems that include Network File System (NFS) mounts, be aware that Ingres:

- Supports NFS client installation configurations in which the DBMS Server process and data directories on one node are accessed by application programs executing on another.

- Does *not* support running DBMS servers on one node and accessing data directories on another network node using NFS. The configuration can cause undetected write errors that lead to database inconsistency.

To check your configuration, type **mount** at the operating system prompt. Make sure that the data directories (II_DATABASE, II_CHECKPOINT, II_JOURNAL and II_LOG_FILE) are not NFS-mounted from a remote node.

For a description of the supported NFS client installation procedures, see the *Installation Guide.*

Recover an Inconsistent Database

The recommended method of recovering an inconsistent database is to use rollforwarddb from VDBA. You can also enter this command from the command line. If no Ingres checkpoint exists, you can recover from an operating system backup.

Make Inconsistent Database Consistent

The recommended way to make your inconsistent database consistent is to use rollforwarddb. It recovers the database from a previous checkpoint and, if journaling was enabled, applies the associated journals. For the full procedure, see the *Database Administrator Guide*.

Use of Operating System Backup

Ingres does not support nor recommend the use of operating system backups as your primary means of ensuring database recoverability. The Ingres checkpoint and journaling programs provide the secure way to ensure that your data is recoverable.

Important! *Operating system backups must be used only as a last resort, when Ingres checkpoints have been lost or destroyed, and only under the direction of technical support.*

When No Backup Exists

If you have an inconsistent database for which no checkpoints or operating system backups exist, you can still gain access to that database and attempt to salvage the data using the verifydb utility.

The verifydb utility can be used to unset the "inconsistent database" flag in the configuration file "aaaaaaaa.cnf". This permits access to the database; however, it does nothing to actually make the data consistent. If verifydb is used to force access to a database that is inconsistent, the state of the database remains unknown. Such a database becomes unsupportable by technical support. Data can be lost and problems occur weeks or months later. Technical support cannot diagnose the state of such a database because the built-in consistency checks have been overridden.

The format "verifydb -oforce_consistent" does not recover a database. It merely allows access and continued operation to a database that is in an inconsistent state.

Gain Emergency Access to an Inconsistent Database Using verifydb

If you *must* use verifydb to gain emergency access to data in an inconsistent database, do so as follows:

1. Save all information, as outlined in What You Need Before Contacting Technical Support (see page 170).

2. Back up the database directory at the operating system level.

3. Run verifydb in report mode by typing the following command at the operating system prompt:

   ```
   verifydb -mreport -sdbname "dbname" -odbms_catalogs -u$ingres
   ```

   Verifydb output is logged in $II_SYSTEM/ingres/files/iivdb.log:

4. To repair inconsistencies in the system catalogs interactively and force the database consistency flag, type the following commands at the operating system prompt:

   ```
   verifydb -mruninteractive -sdbname "dbname" -oforce_consistent -u$ingres
   ```

   ```
   verifydb -mruninteractive -sdbname "dbname" -odbms_catalogs -u$ingres
   ```

5. Call Technical Support if additional assistance is required to resolve the inconsistency.

Performance Problems

Most performance problems stem from multiple causes. A complete performance analysis must include each item in the Diagnostic Hierarchy section below. But if you need to resolve a specific performance problem quickly, focus your attention on that area. The information in this troubleshooting section is designed to assist you in resolving a performance issue after first isolating which factors are influencing performance.

Use the procedures in this section if an Ingres tool "hangs," that is, your program seems to start but nothing happens. Other programs such as SQL display header information, but when you issue a command, nothing happens.

Flow Diagram for Troubleshooting Performance Problems

Use the following flow chart to identify and isolate a system performance problem:
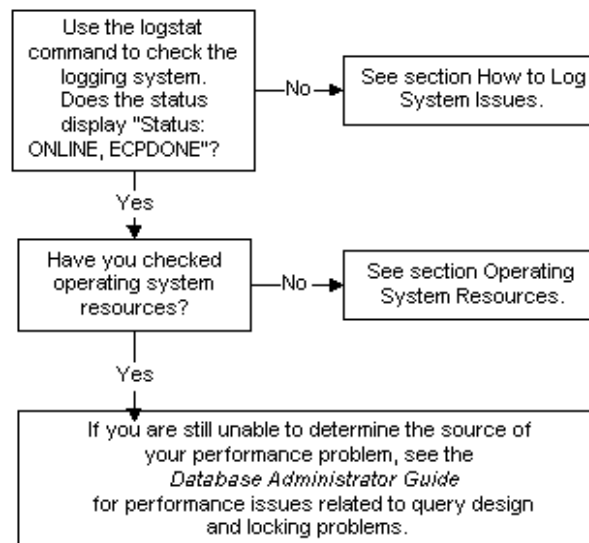
Diagnose DBMS Server Due to Logging System Problem

Use the following procedure to diagnose a DBMS server that is not responding due to logging system problems.

1. Check the logging system by issuing the following command at the operating system prompt to invoke the logstat utility:

   Windows:

   ```
   logstat | more
   ```

   UNIX:

   ```
   logstat | more
   ```

   VMS:

   ```
   define sys$output filename
   logstat
   deassign sys$output
   ```

2. If you are unable to start up logstat, the recovery process (see page 150) has probably taken an exclusive lock and is in recovery.

3. If logstat starts up, check the status field, as described in Logstat Status Fields (see page 160).

Logstat Status Fields

The problem states and resolutions of logstat fields are described in the following table:

Status Field	Description	Action
ONLINE, ECPDONE	A consistency point is completed. The system is fully functional and online.	The logging system has finished a consistency point. This status flag is present most of the time while the logging system is functioning normally. If this is your status, stop this procedure and review Operating System Resources.
LOGFULL	The log file is full.	A status of LOGFULL means that the system is suspended from processing new requests because there is no more room in the transaction log file. It remains so until the problem is corrected. Examine the other status field entries. If ARCHIVE is also indicated, the Archiver is actively processing the

Status Field	Description	Action
		log file to free up space. The LOGFULL condition is removed when the Archiver is finished.
CPNEEDED	A consistency point is needed.	CPNEEDED means the logging system is about to take a consistency point.
ARCHIVE	The Archiver is processing.	The status ARCHIVE means the Archiver process is archiving.
START_ARCHIVER	The Archiver has stopped.	A status of START_ARCHIVER means that the archiver process stopped. Restart the Archiver.
FORCE_ABORT	The force-abort-limit has been reached.	If the status is FORCE_ABORT, a transaction came too close to the end of the transaction log file before the oldest was committed to disk, and reached the force-abort-limit. The oldest open transaction is aborted to free space for the new one.
		You can use logstat (or ipm) to monitor the force abort. The system frees the FORCE_ABORT status when the abort operation is completed.
RECOVER	The recovery process is performing recovery.	If the status is RECOVER, the logging system is in recovery, a normal mode. Recovery requires some time because Ingres is optimized to commit rather than back out of transactions.

Important! *If the server is recovering a lengthy transaction, be aware that shutting down the server with iimonitor's "stop server" (or by killing processes at the operating system level) results in slower recovery. Allow the recovery to proceed normally. You can monitor the progress of the recovery with logstat and iimonitor or the ipm utility*.

Many other normal states appear on the status line. For additional information on reading logstat output, see the Logstat section in the *Command Reference Guide*.

How to Avoid Logfull Abort

To avoid logfull aborts, examine the transaction processing strategy errors that caused the log file to fill. Consider the following points:

- If transactions do many updates or are left open for a long time, you must either commit them more frequently or increase the size of the log file. If the log file size is increased, check the "percentage of logfile written" before a consistency point is taken. For details, see the ipm, logstat, or VDBA output.

- Avoid continuous transaction errors by:

    – Setting autocommit on where applicable

    – Avoiding application errors that are caused by beginning transactions before they get all the user input required or by failing to use Ingres application timeout features. (Timeout features are discussed in the *SQL Reference Guide* and *Forms-based Application Development Tools User Guide.* Also, for a discussion of transaction processing, see the *SQL Reference Guide*.

- If you have taken the above precautions but still get log full/abort, the transaction log is sized too small for the types of transactions you need to process. Read the tips in the next section on choosing the correct transaction log file size.

- Use set log_trace to examine the size and type of log records being written to the log file.

Process of Resizing Transaction Logs

Resizing transaction logs involves these steps:

1. Determine whether the log requires resizing

2. Resize the transaction log file

3. Reestablish Dual Logging

Determine Whether a Log Requires Resizing

To determine whether a log requires resizing, use **one** of the following procedures.

Using VDBA's Performance Monitor, start a monitoring session on the desired node:

1. Select the node in the left pane and click the Monitor toolbar button.

2. Select Log Information in the branch.

3. Click the Header tab in the right pane to display log usage.

Start the Interactive Performance Monitor utility (ipm).

1. Highlight the Log_info menu item and select the Select option.

2. Highlight the Header menu item and select the Select option.

3. The Log file Diagram shows the percentage use of the Transaction Log.

4. Exit the ipm program.

Find the current peak transaction load by using the logstat utility to monitor the amount of log file that is in use during your peak hours. This allows you to choose the correct size for your log file. Because requirements change, you must monitor the log file regularly.

1. Enter the logstat command.

2. Find the value under % of log file in use.

3. If the log file needs to be resized, select the new size.

    UNIX:

    Use mkrawlog to reconfigure a raw log file.

4. Shut down the Ingres installation by entering the ingstop command.

Resize the Transaction Log File

To resize the transaction file, follow these steps:

1. Shut down the Ingres installation by entering the ingstop command.

2. Start the Configuration-By-Forms (cbf) or Configuration Manager (vcbf) utility.

   a. Choose Transaction Log from the menu.

      The file name is displayed along with the current size of the Transaction Log.

   b. Choose Destroy.

      When destroyed, the Transaction Log information table is emptied.

   c. Select the Create option.

   d. Enter the required Transaction Log file size in megabytes.

      The transaction log is created.

3. Exit cbf or vcbf.

4. Restart the Ingres installation by entering the command ingstart.

Important! *Reconfiguring your log file destroys the current contents of the file and leaves an empty, reinitialized log file after the reconfiguration is complete. When Ingres is shut down, all transactions are written to disk; therefore, to prevent inconsistent database problems, reconfigure the log file only after a successful Ingres installation shutdown procedure.*

Reestablish Dual Logging

A failure in one of the two dual log files is most often symptomatic of hardware problems. In the event of such a failure, you must reestablish dual logging at the first opportunity.

To re-establish dual logging after a failure of one of the log files, follow these steps:

1. Shut down the installation.

2. Create a new log file in the location of the failed log.

3. Start the Configuration-By-Forms (cbf) or Configuration Manager (vcbf) utility.

4. Select dual transaction log.

5. Select Reformat.

   This automatically determines which of the log files is valid and copies the contents of the valid log to the newly created log file. After the copy is complete, both log files are marked valid. Dual logging is reestablished.

6. Restart the installation with the ingstart command.

7. Check that both log files are enabled by using Configuration-By-Forms (cbf) or Configuration Manager (vcbf).

Resource and Maintenance Problems

Good performance requires planning and regular maintenance. Make sure your operating system is configured with sufficient resources for Ingres. Insufficient system resources cause poor performance or prevent Ingres from starting.

Tools for Identifying Operating System Problems

The following tools can help you identify operating system resource problems:

- Review the minimum requirements for a basic Ingres installation given in the Readme file. If your particular environment requires more resources, use the Ingres utilities to verify that there are enough resources.

- System resources can be monitored by some operating system utilities. Syntax details are described in Unix Operating System Utilities (see page 177).

  UNIX:

  BSD:

  pstat utility—to display the status of UNIX system tables and system swap space

  vmstat utility—to display virtual memory status

  System V:

  - sar utility—to display activity of various system resources such as CPU utilization, swapping activity, and disk activity.

  - show memory—displays the system memory resources and the amount of non-paged dynamic memory (total, free and in use).

  - show process/id=pid/continuous—displays the amount of page faulting, working set, buffered I/O, and direct I/O the server is doing.

  - show device—indicates if a particular disk drive is out of disk space.

  - show device /files—if there is a problem starting an installation, this command can be used to make sure that an Ingres process is not holding on to a mailbox.

    The installation utility allows the examination of all Ingres installed images, showing the amount of global pages and sections available and used.

  VMS:

  The following VMS tools are very useful for tracing script problems:

  - set verify—allows you to see the commands used in the installation script or any other VMS command procedure.

  - set watch file/class=major (requires CMEXEC VMS privilege)—allows all the files that are being accessed to be displayed. This assists in diagnosing if a location of a file or library is being incorrectly referenced. Use this command when debugging Ingres processes interactively.

  - The displays can be turned off with the following command entered at the operating system prompt:

Important! *This is an unsupported VMS command. Use at your own risk.*

```
set watch file/class=none.
```

Syntax details are described in VMS Operating System Utilities (see page 183).

Diagnose Poor Performance Due to Insufficient System Resources

If Ingres seems slow or unresponsive for no apparent reason, follow these steps to diagnose the problem. Write down any error messages you receive when performing these steps:

1. Connect to your DBMS Server through Ingres monitors:

   a. First display the *server_number* of your DBMS Server using the iinamu utility:

      ```
      iinamu
      IINAMU> show ingres
      ```

   b. Connect to the DBMS Server monitor by typing the command:

      iimonitor server_number

   c. To see the DBMS Server sessions, at the iimonitor prompt type:

      ```
      IIMONITOR> show sessions
      ```

   d. Check the status of the sessions to determine which one is making excessive use of the server. (You can use VDBA to check session status.)

   For syntax details, see the sections iimonitor and iinamu in the *Command Reference Guide*.

2. If repeated "show sessions" commands in iimonitor show that the query session is continually in a CS_EVENT_WAIT (LOCK) state, the problem involves concurrency and locking.

   Alternatively, you can use the VDBA Performance Monitor to check for this problem.

   a. Select Servers in the left pane of the Performance Monitor.

   b. Select INGRES.

   c. Select Sessions in the Servers.

3. If the session alternates between CS_EVENT_WAIT and CS_COMPUTABLE, this indicates that the query is processing. However, if the query is taking an excessive amount of time, set up a trace on it, as described in Trace Utilities (see page 197).

   a. Interrupt the query that is running:

      ■ *Interactively*, use Ctrl+C and wait.

      ■ *In batch or background mode*, use the following command to terminate:

        Windows: In the Task Manager, highlight iidbmst and click on End Process

        UNIX:

        ```
        kill pid
        ```

        VMS:

        ```
        stop process/id = pid
        ```

        where *pid* is the process ID of the query.

        Important! *The command format "stop proc" must be used only as a last resort. Use of this option* can *cause more problems than it solves.*

   b. Issue the command **set gep.**

      ■ Rerun the query. This outputs a query execution plan.

      ■ Alternatively, start an SQL window on the database in VDBA and click the Display Query Execution Plan button to graphically display the query plan.

      ■ It is important to note that interrupting a query requires some time because Ingres is optimized to commit rather than back out of transactions. It takes at least as long to back out of a transaction as to process the transaction normally. The transaction must be fully backed out before sessions can resume and locks are freed.

4. It is useful to note whether the query runs differently when called from other Ingres tools*.* For example, try issuing the same query from Interactive SQL, and Embedded SQL

5. Determine if you can access all of the data in the tables in all components of the query.

   a. From the Terminal Monitor type:

      ```
      select count(*) from tablename
      ```

This verifies that Ingres can sequentially access every row in the table and indicates that other access paths (secondary indexes, hash pointers, B-Tree page pointers, and so on) can cause the problem. Queries using restrictive where clauses probably are using these secondary access methods.

b. Check for permits that apply to this data by typing the following command from the Terminal Monitor:

```
help permit tablename
```

c. Check for restrictions that you have placed on this data through the Knowledge Management Extension features, such as database rules. From the Terminal Monitor enter the following query to determine if a table is subject to restrictions imposed by a database rule:

```
select * from iirules where table_name = 'tablename'
```

Alternatively, in the right pane of VDBA's Database Object Manager (DOM) window:

- Select the Databases branch and the desired database

- Select Tables and the desired table_name

- Select Rules

What You Need Before Contacting Technical Support

Before contacting technical support, gather as much information as possible and have the following information available.

Windows Installations

Save any relevant errors in each of the following logs:

- errlog.log
- iiacp.log
- iircp.log

Have the following information available:

- Your contact ID.
- Your exact Ingres release.
- Your exact operating system release. Obtain this with the VER command.
- The current Ingres installation environment. Use the following command at the operating system prompt:

  ```
  ingprenv > filename
  ```

- The current user environment. Use the following command at the operating system prompt:

  ```
  SET > filename
  ```

- A clear description of what you are trying to do.
- An indication of whether the failure occurs reproducibly.

UNIX Installations

Save any relevant errors in each of the following logs:

- errlog.log
- iiacp.log
- iircp.log

Note: If your system is configured for Ingres Cluster Solution, separate iiacp.log and iircp.log files will exist for each node with _nodename_ appended to the log names. All logs for all nodes must be checked for pertinent errors.

Have the following information available:

- Your Contact ID.

- Your exact Ingres release. Obtain this with the following command at the operating system prompt:

  `cat $II_SYSTEM/ingres/version.rel`

- Your exact operating system release. Obtain this with the following command at the operating system prompt:

  `uname -a`

  Or:

  `cat /etc/motd`

- The current Ingres installation environment. Use the following command at the operating system prompt:

  `ingprenv > filename`

- The current user environment. Use the command:

  BSD:

  `printenv > filename`

  System V:

  `env > filename`

- A clear description of what you are trying to do

- An indication of whether the failure occurs reproducibly

VMS Installations

Have the following information available:

- Your Contact ID.

- Your exact Ingres release including any Ingres patches installed. The following header shows the release number in the format *nn/nn*:

  Ingres Alpha Version *nn/nnnn* (vax.vms/00)

- Your exact operating system release including any VMS patches installed.

- The current Ingres installation environment. To create a file type:

  ```
  define sys$output filename
  show logical II*
  show logical *ING*
  deassign sys$output
  ```

- If you have been unsuccessful in recovering an inconsistent database, it is important to save this additional diagnostic information.

    – For the transactions in the transaction log file, type **logstat**:

    – For the state of the database configuration file, type **infodb** *dbname*

    – For copies of the relevant error log files type:

    ```
    search ERRLOG.LOG date
    type IIRCP.LOG
    ```

    where:

    date is the date the database became inconsistent.

    IIRCP.LOG_*NODENAME* for each node in the cluster, if configured for Ingres Cluster Solution.

    After retrieving all the diagnostic information, type **deassign sys$output**.

- A clear description of what you are trying to do when the problem occurs. This includes commands, queries, and so on.

- An indication of whether the failure occurs reproducibly.

- Save the information in your error logs. Save any relevant errors in:

    – ERRLOG.LOG

    – IIACP.LOG (IIACP.LOG_*NODENAME* for each node in the cluster, if configured for Ingres Cluster Solution)

    – IIRCP.LOG_*NODENAME* for Ingres Cluster Solution

- Save DBMS server parameter settings:

    II_CONFIG:CONFIG.DAT

- A copy of any changes that have been made to II_config:config.dat.

    If the Configuration-By-Forms (cbf) utility is used to update the configuration file, a copy of all changes are written to II_CONFIG:CONFIG.LOG.

- Have Dial-In information available (if allowed).

Chapter 7: Using Monitoring and Tracing Tools

This section contains the following topics:

This chapter describes monitoring and tracing tools that help you troubleshoot the Ingres installation.

Supported Monitoring and Tracing Tools

A number of tools are available to help you troubleshoot and maintain your installation. In addition to the standard commands and utilities, the tools include:

- Special purpose utilities:

    - Ingres system utilities

    - Operating system utilities

- Trace utilities:

    - Ingres trace utilities

    - Operating system trace utilities

Note: You can use Ingres Visual Manager, Visual Performance Monitor, and VCDA to visually monitor and maintain your installation. For additional information about these tools, see the chapter "Managing Your System and Monitoring Performance" and online help.

System Utilities

Ingres system utilities monitor aspects of the installation for troubleshooting or other special purposes. The system utilities are not needed for normal operation and are generally used with the help and advice of Technical Support. For more information about system utilities see the *Command Reference Guide.*

Day-to-day Ingres commands follow:

	Command	Name	Description
	cacheutil	Shared Buffer Cache Utility	Lists status information on active shared memory segments; can also be used to deallocate an inactive shared memory buffer cache
Windows:	ipcclean	Shared Memory Allocation	Deallocates shared memory ▧
UNIX:	csinstall, cscleanup, ipcclean csreport	Shared Memory Allocation	Allocates, deallocates, and lists status of UNIX shared memory and semaphore resources. Allocation is based on the value assigned II_LG_MEMSIZE. ▧
	iimonitor	Server Maintenance Utility	Lists DBMS server information; also allows a privileged user to stop the server or an active session
	iinamu	Name Server Utility	Lists current servers; also allows a privileged user to add and delete servers
VMS:	iishowres	Show Resources	Lists shared memory used by the locking and logging system ▧
	lartool	Logging, archiving and recovery tool	Allows the system administrator to manually abort or commit a distributed transaction involved in two-phase commit protocol
	lockstat, logstat	Locking, Logging Status	Display information from the locking or logging system. These functions are replaced by the forms-based ipm utility.

Command	Name	Description
rcpconfig	Logging/ Locking Configuration	Manipulate log files or shut down the logging/locking systems (RCP module)
ija	Logging, archiving, and recovery tool	Allows the system administrator to manually abort or commit a transaction involved in two-phase commit protocol

Operating System Utilities

Many operating systems provide utilities to administer and troubleshoot. Most of the operating systems provide context sensitive help to use the tools.

Windows Operating System Utilities

You can use the following Windows operating system utilities to monitor Ingres:

- Windows Diagnostics
- Windows Performance Monitor
- Windows Event Viewer
- Windows Registry Editor
- Windows Task Manager

For a full description of the Windows utilities, see your Windows documentation.

Windows Diagnostics

The Windows Diagnostics program can help you determine your operating system's configuration. This tool can be found in Settings, Control Panel, Administrative Tools, Computer Management.

Windows Performance Monitor

Performance Monitor is a Windows graphical tool for measuring the performance of your own computer or other computers on a network. On each computer, you can view the behavior of objects such as processors, memory, cache, threads, and processes. Each of these objects has an associated set of counters that provide information on such things as device usage, queue lengths, and delays, as well as information used for throughput and internal congestion measurements.

It provides charting, alerting, and reporting capabilities that reflect current activity along with ongoing logging. You can also open log files at a later time for browsing and charting as if they were reflecting current activity. To monitor performance on Windows, see the operating system documentation.

Windows Event Viewer

Event Viewer is a tool for monitoring events in your system. You can use Event Viewer to view and manage System, Security, and Application event logs. To access the Event Viewer, right-click on the My Computer icon and select Manage. The Computer Management Window is displayed. The Event Viewer is available under the System Tools.

Windows Registry Editor

This program can be used to view the system configuration and environment. For a description of how the information is presented and the capabilities of the utility, see the online help.

To start the Registry Editor

1. Click Start, Run.

2. Type **REGEDT32** in the Run dialog. (Alternative: Type **REGEDT32** in a command window.)

   The Registry Editor opens.

Windows Task Manager

The Task Manager enables you to monitor and control your computer and what is running on it. It shows you programs and processes that are running as well as performance. To access the Task Manager, right click an empty area in the task bar and click Task Manager.

UNIX Operating System Utilities

You can use the following UNIX operating system utilities to monitor Ingres:

- ps

- iostate

- pstat

- sar

- sdysdef

- vmstat

For a full description of the UNIX options, see your UNIX documentation (or online help). A detailed description of the utilities can be found in the *Command Reference Guide*.

Note: Not all of the utilities are present on every UNIX system. Some are present only in a BSD or System V environment but not both.

ps

This command provides virtual memory and cpu information on each active process submitted from your account. Here is a sample ps output:

```
PID   TT  STAT  TIME  SL  RE  PAGEIN  SIZE  RSS   LIM  %CPU  %MEM  COMMAND
xx06  p3  s     28:50 13  99  45886   3696  2856  xx   0.0   39.3  iidbms
xx94  p3  s      4:24  0  99   2899    720   344  xx   0.0    4.7  dmfrcp
xx09  p3  I      0.51 99  99   4488      4   184  xx   0.0    2.5  iislave
xx19  p3  I      0.57 99  99   5764     64    17  xx   0.0    2.4  iislave
xx96  p3  I      0.04 99  99   1852    696   160  xx   0.0    2.2  dmfacp
```

The display fields are as follows:

Field	Description
PID	Process ID field
TT	Controlling terminal
STAT	Process status
	Runnability of the process: Runnable (r), Stopped (t), Disk or other short-term wait (d), Sleeping (s), or Idle (i)
	Swap status: Swapped out (w) or Loaded in core (blank)
	Process priority change: Reduced (n), Increased (>) or No change (blank)
SL	Sleep time (seconds blocked)
RE	Residency time (seconds in core)

Field	Description
PAGEIN	Number of disk I/Os resulting from page references not in core
SIZE	Virtual process size
RSS	Resident set size
LIM	Soft memory limit (setrlimit), else "xx"
%CPU	CPU utilization (1 minute decaying average)
%MEM	Memory utilization
COMMAND	Process name

iostat

The iostat command returns information about I/O status. It lists statistics on current I/O activity for each disk device and system CPU utilization percentages. Here is a sample iostat output:

```
tty cpu
tin    tout    us    ni    sy    id
  1      18    19     0     3    78


/dev/*dsk/c0d*s*/dev/*dsk/cld*s*/dev/*dsk/c2d*s*/dev/*dsk/c4d*s*
bps    sps    msps  bps    sps    msps  bps    sps    msps  bps    sps    msps
 2     0.1    61.7   1     0.0    95.5   1     0.0    60.4   2     0.2    44.3
```

- The tin and tout display fields show the number of characters written to and from terminal devices.

- CPU information includes the % time spent in user mode (us), "niced" user mode (ni), system mode (sy), and idle (id).

- The disk I/O for each disk device shows the average number of blocks transferred per second (bps), average number of seeks per second (sps), and average time per seek in ms (msps).

pstat (BSD)

Print Statistics utility is useful for examining system tables and system swap space.

The command **/**etc/pstat –s, shows system swap space activity. An example output:

```
589944 used, 667364 free, 61864 wasted
avail (num*size): 316*2048 5*1024 6*512 13*256 21*128 38*64 54*32
```

Note: For a description of the output units, see your operating system guide or online MAN pages. On some machines output is in bytes (as in this example). On others it is in units equal to the machine's page size.

The command /etc/pstat -T gives a list of used and free slots in various system tables. Here is a sample output:

```
1717/3216 files
937/2384 inodes
333/1300 processes
266/325 mfiles
595176/1256780 swap
```

The pstat utility can be used to give other useful information if the user has knowledge of the UNIX kernel. It can tell you, for example, the state of a running process, which files are open, which operating system locks apply and which processes have which files open.

sar (System V)

The sar (System Activity Reporter) command can be used to list CPU utilization statistics, disk and swapping activity, and other status options. The UNIX sar command samples cumulative activity counters in the operating system and reports on various system activities. Just a few of the reports available that are of use for the system administrator are CPU utilization, disk activity, and swapping activity. See examples below.

There are many other options available, including reports on unused memory pages and disk blocks (swap space), and message and semaphore activity. For the system administrator, sar is an indispensable tool.

- The command **sar -u 5**, shows CPU utilization output as in the following sample output:

```
06:38:26   %usr   %sys   %wio   %idle
06:38:31      1      3     96       0
```

This output shows the portion of CPU time spent running in user mode, system mode, idle with some process waiting for block I/O, or otherwise idle.

- The command **sar -d 5**, shows disk activity output as in the following sample output:

```
06:39:53 device   %busy   avque   r+w /s   blks/s   avwait   avserv
06:39:53 disc0-0     16    13.6        6       17    349.6     29.7
         disc0-5      0     1.0        0        2      0.0     20.0
```

The display shows the portion of time the device was busy servicing requests (%busy), the average number of outstanding requests (avque), the number of data transfers (r+w/s), the number of bytes transferred in 512-byte units (blk/s), the average time in milliseconds that requests wait idly on queue (avwait), and the average time for requests to be serviced (avserve).

- The command **sar -w 5** shows swapping activity output as in the following sample output:

```
06:40:45   swpin/s   bswin/s   swpot/s   bswot/s   pswch/s
06:40:50      0.00       0.0      0.00       0.0        19
```

The fields swpin/s, swpot/s are the number of swap transfers. The fields bswin/s, bswot/s are the number of 512-byte units transferred in/out. The field pswch/s is the number of process switches.

sysdef (System V)

The /etc/sysdef (System Definition) command (implemented only on some System V operating systems) displays a list of all tunable kernel parameters and their configuration in the currently active kernel. Here is a sample sysdef output:

```
Tunable Kernel Parameters:

  250    buffers in buffer cache (NBUF)
   30    entries in callout table (NCALL)
  150    inodes (NINODE)
  150    s5inodes (NS5INODE)
  150    entries in file table (NFILE)
   25    entries in mount table (NMOUNT)
  100    entries in proc table (NPROC)
  210    entries in shared region table (NREGION)
  120    clist buffers (NCLIST)
   25    processes per user id (MAXUP)
   64    hash slots for buffer cache (NHBUF)
   20    buffers for physical I/O (NPBUF)
   50    size of system virtual space map (SPTMAP)
   16    fraction of memory for vhandlow (VHNDFRAC)
    0    maximum physical memory to use (MAXPMEM)
   10    auto update time limit in seconds (NAUTOUP)
   60    maximum number of open files per process (NOFILES)
  256    number of streams queues (NQUEUE)
   48    number of streams head structures (NSTREAM)
    4    number of 4096 bytes stream buffers (NBLK4096)
   32    number of 2048 bytes stream buffers (NBLK2048)
   32    number of 1024 bytes stream buffers (NBLK1024)
   32    number of 512 bytes stream buffers (NBLK512)
   64    number of 256 bytes stream buffers (NBLK256)
  128    number of 128 bytes stream buffers (NBLK128)
  256    number of 64 bytes stream buffers (NBLK64)
  256    number of 16 bytes stream buffers (NBLK16)
  128    number of 4 bytes stream buffers (NBLK4)

 2560    maximum size of user's virtual address space in pages (MAXUMEM)
 2560    for package compatibility equal to MAXUMEM (MAXMEM)
   25    page stealing low water mark (GPGSLO)
   40    page stealing high water mark (GPGSHI)
    9    page aging interval (AGEINTERVAL)
    1    bdflush run rate (BDFLUSHR)
   25    minimum resident memory for avoiding deadlock (MINARMEM)
   25    minimum swappable memory for avoiding deadlock (MINASMEM)
    1    maximum number of pages swapped out (MAXSC)
    1    maximum number of pages saved (MAXFC)
```

```
 *
 * Utsname Tunables
 *
      3.2  release (REL)
chipmunk  node name (NODE)
chipmunk  system name (SYS)
        2  version (VER)




*
* Streams Tunables
*
   87  number of multiplexor links (NMUXLINK)
    9  maximum number of pushes allowed (NSTRPUSH)
  256  initial number of stream event calls (NSTREVENT)
    1  page limit for event cell allocation (MAXSEPGCNT)
 4096  maximum stream message size (STRMSGSZ)
 1024  max size of ctl part of message (STRCTLSZ)
   80  max low priority block usage (STRLOFRAC)
   90  max medium priority block usage (STRMEDFRAC)




*
* RFS Tunables
*
   10  entries in server mount table (NSRMOUNT)




*
* IPC Messages
*
  100  entries in msg map (MSGMAP)
 2048  max message size (MSGMAX)
 4096  max bytes on queue (MSGMNB)
   50  message queue identifiers (MSGMNI)
    8  message segment size (MSGSSZ)
   40  system message headers (MSGTQL)
 1024  message segments (MSGSEG)
*
* IPC Semaphores
*
   10  entries in semaphore map (SEMMAP)
   10  semaphore identifiers (SEMMNI)
   60  semaphores in system (SEMMNS)
   30  undo structures in system (SEMMNU)
   25  max semaphores per id (SEMMSL)
   10  max operations per semop call (SEMOPM)
   10  max undo entries per process (SEMUME)
32767  semaphore maximum value (SEMVMX)
16384  adjust on exit max value (SEMAEM)
```

```
*
* IPC Shared Memory
*
 1048576 max shared memory segment size (SHMMAX)
       1 min shared memory segment size (SHMMIN)
     100 shared memory identifiers (SHMMNI)
       6 max attached shm segments per process (SHMSEG)
     512 max in use shared memory (SHMALL)
*
* File and Record Locking
*
    100 records configured on system (FLCKREC)
```

vmstat

The vmstat (Virtual Memory Statistics) command returns virtual memory status information, including process states and paging activity. Here is a sample vmstat output:

```
procs      memory            page      disk       faults     cpu
r b w avm  fre di  re rd pi po de z0 z1 z2 z3 in sy cs us sy id
1 0 0 2536 456 24  2  1  4  0  0  1  1  0  1  24 475 23  4  6 91
0 0 0 2748 356 24  0  0  0  0  0  2  0  0  3  26 323 29  1  5 95
0 0 0 1044 344 24  0  0  0  0  0  0  0  0  0  16 216 18  0  3 97
0 0 0 2288 344 24  0  0  0  0  0  0  0  0  0  19 334 27  1  4 95
1 0 0 2372 332 24  0  0  4  0  0  0  0  0  1  28 552 40  1  6 93
```

The procs columns define the process states: in run queue (r), blocked for resources (b), and runnable or short sleeper (w).

The memory columns show virtual and real memory status: avm is active virtual pages (belonging to processes active in approximately the last 20 seconds), fre is size of the free list, and di is the number of dirty pages.

The page columns show page faults and paging activity. These are expressed in units per second, averaged over 5 seconds: re are page reclaims, rd are page reclaims from the dirty list, and pi, po are pages paged in/out. The de field is anticipated short-term memory shortfall.

The disk columns list disk activity, showing the device name and operations per second.

VMS Operating System Utilities

VMS utilities vary depending on the operating system release. For detailed information on utilities supported by your system, see your operating system guides (or online help).

help

The VMS online help facility (help) provides information on the specified command. The syntax is help *command*.

monitor

The VMS monitor utility returns dynamic information about VMS system performance for a specified component. Some of the most useful components to monitor are:

- monitor system — monitors system statistics

- monitor process/topcpu—monitors the process currently using the most CPU

- monitor process/topdio—monitors the processes that use the most disk I/O

- monitor process/topbio—monitors the processes that use the most buffered I/O

- monitor process/topfault— monitors the processes that cause the most page faults

- monitor pool—monitors non-paged pool statistics

show

The VMS show utility returns process and other VMS system status information on a specified component. Qualifiers (**/**qualifier) are used on certain components, as shown in the examples that follow. Some useful forms of the VMS show command appear in the following list.

- **show system or show system/full**—displays status information about current processes. Here is a sample output:

```
date and time stamp
process ID number
process name and identification with UIC
processing state
priority
total process I/O
cumulative cpu time used
cumulative page faults
amount of physical memory being used
type of process

VAX/VMS V6.0 on node FOO 9-FEB-2001 09:34:17.48  Uptime 13 13:46:12
Pid        Process Name        State   Pri   I/O     CPU            Page flts   Ph.Mem
00205374   DMFRCPBE            HIB     6     6270    0 00:00:15.71   2756        4280
00202F75   DMFACPBE            HIB     5     934     0 00:00:57.09   1115        2302
00204376   II_GCNBE            HIB     6     667     0 00:00:03.15   782         1363
00202B77   II_GCC_BE_2B77      HIB     5     142     0 00:00:02.15   664         1549
00205578   II_DBMS_BE_5578     HIB     5     38540   0 00:04:31.58   9390        13903
002002C3   DMFCSPBE            HIB     5     71071   0 00:00:22.27   399         173
```

- **show system /out = filename**—outputs to a file. This provides a record of the current status.

- **show process**—displays information about a process and subprocesses. This command requires the GROUP privilege to show other processes in the same group or the WORLD privilege to show processes outside your group. If no qualifier is entered, only a basic subset of information is displayed:

```
date and time stamp
process terminal
user name and UIC
node name
process name and process identification
priority
default directory
allocated devices
```

  Some of the show process qualifiers you can use are:

  /memory—displays the process' use of dynamic memory areas. This qualifier is allowed only for the current process.

  /quotas—displays, for each resource, a quota or a limit. The quota values reflect any quota reductions resulting from subprocess creation. Limit values reflect the resources available to a process at creation.

  /priv—displays current privileges for the process.

- **show cluster**—monitors and displays cluster activity and performance.

- **show cpu**—displays the current state of the processors in a VMS multiprocessing system. This command requires a change of mode to the kernel (CMKRNL) privilege.
  The show cpu qualifiers are:

  /all—selects all configured processors, active or inactive, as the subject of the display.

  /full—produces information from the summary display and also lists the current CPU state, current process (if any), revision levels, and capabilities for each configured processor. The display indicates which processes can execute only on certain processors in the configuration.

- **show device**—displays the status of a device on the system. You can use the following show device qualifiers:

  /allocate—displays all devices currently allocated to processes.

  /file—shows all open files on the device, together with the process name and the process ID that opened the file.

  /mounted—displays all devices currently having volumes mounted.

- **show logical /full**—displays all logical names in one or more logical name tables, or displays the current equivalence strings assigned to specified logical names, with iterative translations.

- **show memory /full**—displays status information on VMS system tables and system swap space.

- **show symbol**—displays the current value of a local or global symbol. Ingres symbols are defined by running the following:

  II_SYSTEM:[INGRES.UTILITY]INGSYSDEF.COM – Defines Ingres system administration, DBA, and user symbols.

  II_SYSTEM:[INGRES.UTILITY]INGDBADEF.COM – Defines Ingres DBA and user symbols.

  II_SYSTEM:[INGRES.UTILITY]INGUSRDEF.COM – Defines Ingres user symbols.

- **show users**—displays status information on current users.

- **show working_set**—displays the working set limit, quota, and extent assigned to the current process.

sysgen

The VMS sysgen utilitty displays system configuration parameters. To run this utility, execute the following command at the operating system prompt:

```
run sys$system:sysgen
```

For a full description of the system definitions, see your VMS System manual.

Vendor Utilities

Many vendors also provide monitor utilities. These often present a more user-friendly interface than the generic utilities mentioned above. Check your operating system documentation for availability of these useful vendor-specific utilities.

Error Messages

Errors in Ingres can be classified into Fatal and Non-Fatal errors. Ingres displays error messages in a particular format. This format helps you find the specific problem that Ingres has encountered. Error messages and their formats are explained in the following sections.

Error Message Format

Error messages in Ingres consist of a three-part error code and accompanying message text. The code segment has the format:

```
E_FCxxxx_msg message text
```

The message codes are as follows:

E_, I_, or W_

E_ (error message), I_(informational message), or W_ (warning message).

FC

Two-letter Facility Code (see page 188).

xxxx

4-digit hexadecimal number that represents the value returned to the application for use in error handling. The 2-letter facility code combined with the 4-digit hexadecimal number uniquely identifies this error code.

_msg

"*msg*" is an abbreviated message text or decimal number. As a decimal, it is always equivalent in value to the preceding hexadecimal number.

Message Help Files

Help is available online to aid in reading error messages. The error messages are stored in a single file messages.txt in the directory $II_SYSTEM/ingres/files/english/messages/.

Fatal Errors

Fatal errors are those errors that require correction before Ingres can proceed with the program. All errors, including fatal errors, are recoverable.

Fatal errors occur in the following categories:

- Ingres tools, such as QBF or ABF

- Ingres server errors

  – Environmental OS/hardware limits and problems

  – A command limit is reached, for example, exceeding the maximum number of aggregates in a query

  – System software error

To remove the cause of a fatal error, use the troubleshooting techniques in the chapter Troubleshooting Ingres.

Non-Fatal Errors

Non-fatal errors are diagnostic indications of user errors, usually in application development. For example, a non-fatal error code is returned when a user attempts to destroy a non-existent table. These errors are not severe and do not halt the user program or Ingres. User applications can choose to print or suppress these errors.

Ingres Facility Codes

Ingres facility codes consist of a two-character code that is generated by a particular facility. For example, the Abstract Data Type Facility has a facility code of AD so error messages generated by this facility have an error message format of:

`E_AD`*`xxxx_msg`*

Error messages often include an explicit reference to which internal Ingres function returned the error. A list of these modules and their functions can help you identify the source of errors requiring troubleshooting.

Facility Codes for Primary Components

The facility codes for the primary Ingres components, such as the DBMS Server and General Communications Facility, are shown in the following table:

Facility Code	Facility
AD	Abstract Data Type Facility (ADF)
DM	Data Manipulation Facility (DMF)
DU	Database Utility Facility (DUF)
GC	All General Communication Facilities (GCF), including: ■ Communications Server (GCC) ■ General Communications Area (GCA) ■ Name Server (GCN)
OP	Optimizer Facility (OPF)
PS	Parser Facility (PSF)
QE	Query Execution Facility (QEF)
QS	Query Storage Facility (QSF)
RD	Relation Description Facility (RDF)
SC	System Control Facility (SCF)
US	User Errors

Server-Only Components

The server-only components are mostly DMF routines. The facilities listed in the following table are all subcomponents for Compatibility Library (CL) and do not appear in the facility-code part of an error message. All messages involving these facilities start with E_CL, and usually contain the two-letter facility code in the abbreviated message part of the error text.

Facility Code	Name	Description	Used By
CS	Central System	Control of server	SCF, Utilities
DI	Database I/O	Access to Database files (different from SI)	DMF, Utilities
JF	Journal Files	Access to Journal files (different from DI)	DMF, Utilities

Facility Code	Name	Description	Used By
LG	Logging	Server logging (transaction log)	DMF
LK	Locking	DBMS data locking	DMF, Utilities
SR	Sorting	Sorting for queries	DMF
TR	Tracing	DBMS event tracing	Server components

Tools-Only Components

The Ingres Tools-only components are shown in the following table:

Facility Code	Name	Description	Used By
AR	ABF Runtime	Application-By-Forms runtime component	ABF
DY	Dynamic Utilities	Dynamic linking	ABF
OL	4GL Support	Produce Ingres 4GL object code	ABF
PC	Process Control	Fork, sleep, exit, and so forth	Tools components
PE	Permissions	Object access authority	Tools components
TE	Terminal Driver	Terminal I/O	Tools terminal driver
UT	Utility Invocation	Compile, link, call, print, and so forth	Tools components

Components for Stream File Management

Ingres components for stream file management are shown in the following table. The facilities listed below are all subcomponents for Compatibility Library (CL) and do not appear in the facility-code part of an error message. All messages involving these facilities start with E_CL, and usually contain the two-letter facility code in the abbreviated message part of the error text.

Facility Code	Name	Description	Used By
CP	Switch protections	Special-purpose file permissions handling	auditdb

Facility Code	Name	Description	Used By
LO	Locations	Abstract file system support (hierarchical)	Tools components
SI	Stream I/O	Stream functions	Tools components

Components for Utility Routines

Ingres components for utility routines are shown in the following table:

Facility Code	Name	Description	Used By
CI	Authorization	Authorization string interpretation	Server and Tools components
ER	Error Messages	Error Message Handling/supports international	Server and Tools components
EX	Exception Handling	Establish routine chaining	All
GC	GCA CL	GCA (IPC) support	All
GV	Global Variables	Global symbols (Release ID, date, and so forth)	All
ID	Object IDs	User identifications	Usually Tools components
ME	Memory Allocation	Memory management	Usually Tools components
NM	Logical Symbols	Logical symbol management (or simulation)	All
QU	Queue Manipulation	Management of queues	All
TM	Timing	Timing services	All; some timing is server only

Miscellaneous Components

Miscellaneous components are shown in the following table:

Facility Code	Name	Description	Used By
BT	Bit Manipulation	Utility routines to test, set, clear bits	All
CM	Character Manipulation	Character handling and for Kanji support	All
CV	Conversion	Convert internal representations	All
MH	Math	Functions (tan, ln, log, exp, and so forth)	All
ST	String Handling	Utilities for string operations	All

Log Files

Ingres maintains log files to which it writes information about the installation activities.

Transaction Log File

Each installation has a transaction log file and an optional dual log file. The log file holds information about all open transactions and is used to recover active databases after a system failure. Optionally, you can change its size and number of partitions at startup.

UNIX: The UNIX log file can be created as a raw partition or a number of raw partitions.

Error Log

The main error log is a readable text file that you can use for troubleshooting. The error log file (errlog.log) is maintained in $II_SYSTEM/ingres/files/ directory.

Messages about the installation are appended to this log with the date and time at which the error occurred. This is generally the first place to look when troubleshooting a problem.

The error log contains the following information:

- Error messages

- Warning messages

- Archiver shutdown

- DBMS server start up and shutdown

The system administrator maintains the error log file. The file continues to grow until manually truncated. The installation must be shut down before truncating or removing the errlog.log file.

Archiver Log

The archiver log (iiacp.log) contains information about the current archiver process. The archiver log file is appended to when the archiver process starts. The log contains the following information:

- Archiver start up

- Archiver error messages

- Archiver warning messages

The archiver log is located in the $II_SYSTEM/ingres/files/ directory.

Recovery Log

The recovery log (dmfrcp) contains information about the current recovery process. The recovery log file (iircp.log) is located in the $II_SYSTEM/ingres/files/ directory.

This file is appended to when the recovery process starts. The log contains the following information:

- Current logging and locking parameter values

- Recovery process error messages

- Recovery process warning messages

- Recovery operations information

The recovery log must be monitored if you are unable to connect to Ingres and suspect that the DBMS Server is in recovery mode.

Primary Configuration Log Files

Ingres maintains transcripts of various configuration operations. The primary configuration log files that Ingres uses are as follows:

config.log

The config.log file contains a log of the changes made with the Configuration Manager or Configuration-By-Forms (cbf) utility.

rcpconfig.log

The rcpconfig.log file contains log and error information of the last time rcpconfig was run. For details of the rcpconfig command, see the *Command Reference Guide*.

The primary configuration log files are located in the $II_SYSTEM/ingres/files/ directory.

Optional Configuration Log Files

Ingres provides the following optional configuration log files. These files, if present, are in the directory indicated by II_CONFIG.

Note: In UNIX, the II_CONFIG file must always be in $II_SYSTEM/ingres/files.

iilink.log

Contains a log of the last time the iilink command was run. This log is used only in conjunction with the Object Management Extension.

Note: This log exists on Windows only.

iivdb.log

Contains a transcript of the last time verifydb was used to diagnose or attempt recovery of a damaged or inconsistent database. This file is created the first time verifydb is run.

Note: This log does not exist on Linux.

lartool.log

Contains a transcript, with any errors, of the last time lartool was used to manually abort or commit a running transaction. This file is created the first time lartool is run.

Other Optional Log Files

The following types of optional log files are present on your installation:

- Individual process logs. For the following processes you can set up a separate log file to isolate the error messages relating to that process:

  - DBMS

  - GCC

  You normally do this only for a specific troubleshooting purpose. The default is that these messages are sent to the errlog.log. If you define one of these separate error logs, all messages are sent to *both* that file and errlog.log.

- Trace log for tracking messages at a greater level of detail

- Log for an optional Ingres facility

These optional log and trace log files can be established by setting the associated Ingres variables:

DBMS Error log

The DBMS error log, optionally defined as a separate file.

This log file is established by setting the Ingres variable II_DBMS_LOG to a user file name For additional information, see the appendix "Environment Variables and Logicals." The default is all DBMS server errors and messages are sent to errlog.log.

GCC trace log

The GCC trace log is set up for specific troubleshooting efforts. You set II_GCA_LOG to a user file name. The associated Ingres variable II_GCC_TRACE defines the level of tracing. These Ingres environment variables/logicals are listed in the appendix "Environment Variables/Logicals." Details on using a GCC trace are descried in User-Server Communications (see page 208).

Star error log

The Star error log for installations running Ingres Star, if defined.

This log file is established by setting the Ingres variable II_STAR_LOG to a user file name. For additional information, see the appendix "Environment Variables and Logicals." The default is all Ingres Star errors and messages are sent to errlog.log.

Trace Utilities

Commands are available to trace such things as disk file access, locks, user interface-to-DBMS server communications, query plans of the Ingres Query Optimizer and various types of memory usage. For more information on tracing, see the *Database Administrator Guide*.

DBMS server tracing is generally enabled by some form of the set command, although some forms of tracing make use of Ingres environment variables and logicals. Set commands used specifically for debugging and troubleshooting are discussed in this document. More general set commands are documented in the *SQL Reference Guide*.

ODBC Call-level Interface

The Ingres ODBC Call-level Interface (CLI) follows the existing convention of ODBC tracing as performed by the UNIX ODBC Driver Manager and Microsoft ODBC Driver Manager. Registry or configuration files are scanned for trace and trace log settings. Tracing output is similar to what is currently provided by the Ingres ODBC tracing DLL on Windows. Optionally, an application can set standard ODBC tracing using the following Ingres environment variables:

II_ODBC_LOG

A string indicating the path and file name of an ODBC trace file. For example, the path and file name of an ODBC trace file for UNIX and Linux is \tmp\odbc.log and for Windows is c:\temp\odbc.log.

II_ODBC_TRACE

A positive integer with a value of 1, 3, or 5. A setting of 1 provides standard ODBC tracing and is the most useful for debugging ODBC CLI applications. A setting of 3 includes ODBC function entry calls in the Ingres ODBC driver (as opposed to the ODBC CLI driver manager). A setting of 5 displays information about internal function entries in the Ingres ODBC driver.

If an ODBC application does not use the Ingres ODBC CLI, the ODBC trace settings of 3 and 5 are still recognized, but the setting of 1, which belongs to the driver manager component, is not. However, the existing tracing capability for third-party driver managers is unaffected. In this case, the driver manager tracing is written to the ODBC trace log as specified in the registry (Windows) or odbcinst.ini file (UNIX and Linux), but the detailed tracing information is written to the log file as specified by the II_ODBC_LOG environment variable.

Set Statement

The SET statement is used to specify a number of runtime options and has the general format:

```
set [no]option [additional parameters];
```

The *option* is set on or active by SET *option* or SET *option* ON (depending on the option). It is turned off by a corresponding SET NOOPTION or SET *option* OFF. Additional parameters are required, depending on the option. For a detailed syntax description of the SET statement, see the *SQL Reference Guide.*

Environment Variables and Logicals Commonly Used with Set

SET statements are executed as part of query language startup procedures at selected levels of scope. A SET statement is entered directly at a terminal for temporary settings or executed using an Ingres environment variable/logical to establish a default option setting for all users. Several levels are shown in the following table:

Where Specified	Scope of Effect
ING_SYSTEM_SET	Executed whenever Ingres is started up, affecting all users
ING_SET	Executed whenever any Ingres tool connects to a server
ING_SET_DBNAME	Affects only Ingres tools starting up on the specified database
II_EMBED_SET	Similar to ING_SET, but diagnostic output is to a file
SET statement entered directly	In effect until changed by a subsequent SET statement for the option. This occurs by another direct SET statement or using any of the Ingres environment variables and logicals above reestablishing the default.

Example: Set Statement Entered Directly

One or more set options can be specified using the _SET environment Ingres environment variables and logicals. They can be assigned directly from the operating system shell. Separate multiple SET statements by semicolons (up to a limit of 64 characters), as shown in the following examples:

Windows:

```
SET ING_SET=set nojournaling;set printqry
```

UNIX:

C Shell:

```
setenv ING_SET "set nojournaling; set printqry"
```

Bourne Shell:

```
ING_SET = "set nojournaling; set printqry" export ING_SET
```

VMS:

```
DEFINE ING_SET "SET NOJOURNALING; SET PRINTQRY"
```

Example: Set Statements in an Include File

SET statements can also be implemented by means of an INCLUDE statement. The INCLUDE statement allows you to place the SET statements in a file and specify the file name in the setenv (or equivalent) command. Use of the INCLUDE option also avoids the 64-character line limit.

For example:

Windows:

```
set ING_SET=include c:\extra\ingres\set.ing
```

UNIX:

C Shell:

```
setenv ING_SET "include /extra/ingres/set.ing"
```

Bourne Shell:

```
ING_SET = "include /extra/ingres/set.ing"
export ING_SET
```

VMS:

```
DEFINE ING_SET -"INCLUDE DUA0:[EXTRA.INGRES]SET.ING"
```

The file *set.ing* includes the following statements:

```
set autocommit on;
  set lockmode on mastertable
  where level = page,
  maxlocks = 10;
```

Set Statements in Startup Files

SET statements are automatically executed as part of query processor start up. Various startup defaults are enabled either by setting an Ingres environment variable/logical (such as ING_SET or II_EMBED_SET described above) or by including these SET statements in one of the query processor startup files.

For an alphabetical listing of Ingres environment variables/logicals, see the appendix "Environment Variables and Logicals."

Set Options for Tracing Queries

SET options of special interest in query troubleshooting include:

- [NO]PRINTQRY—Set query display on or off.
- [NO]QEP—Set query execution plan display on or off.
- JOINOP [NO]TIMEOUT—Set optimizer timeout on or off.

The set printqry Option

The command SET PRINTQRY prints out the query before it is optimized and executed. This is especially useful when evaluating queries for performance tuning or troubleshooting.

Example: Turn Query Display On Using ING_SET

The following command turns query display on:

Windows:

```
SET ING_SET=set printqry
```

UNIX:

C Shell:

```
setenv ING_SET "set printqry"
```

Bourne Shell:

```
ING_SET = "set printqry"
export ING_SET
```

VMS:

```
DEFINE ING_SET "SET PRINTQRY"
```

As an example, assume that QBF is started up. The following is displayed:

```
Query Buffer:
set Autocommit on
Query Parameters:

Query Buffer:
select cap_value, cap_capability from iidbcapabilities
Query Parameters:

Query Buffer:
select user_name, dba_name from iidbconstants
Query Parameters:
```

Example: Turn Query Display On Using II_EMBED_SET

The following command turns query display on through II_EMBED_SET. This is similar to the ING_SET environment variable/logical. All information is gathered in the client application and printed to the log by the library routines linked to the client application:

Windows:

```
SET II_EMBED_SET=printqry
```

UNIX:

C Shell:

```
setenv II_EMBED_SET "printqry"
```

Bourne Shell:

```
II_EMBED_SET = "printqry"
export II_EMBED_SET
```

VMS:

```
DEFINE II_EMBED_SET "PRINTQRY"
```

The query output with query timings is placed into a file called iiprtqry.log. Here is a sample output:

```
Query text:
select * from iirelation
Query Send Time:  Thu Mar 26 17:20:43 2001
Query Response Time: Thu Mar 26 17:20:44 2001
Response CPU Time: 2630
Qry End CPU Time: 3370
```

This type of query monitoring can be useful for spotting slow-running queries in applications.

The set qep Option

The SET QEP option provides a display of the optimizer's query execution plan (QEP) for the query after it has been optimized but before it is executed.

The following command turns QEP display on as the default ING_SET level:

Windows:

```
SET ING_SET=set qep
```

UNIX:

C Shell:

```
setenv ING_SET "set qep"
```

Bourne Shell:

```
ING_SET = "set qep"
export ING_SET
```

VMS:

```
DEFINE ING_SET "SET QEP"
```

The Ingres Query Optimizer and QEPs are described in detail in the *Database Administrator Guide.*

The set joinop notimeout Option

The SET JOINOP NOTIMEOUT option can be used in tracing query performance. This statement turns the optimizer timeout feature off. With timeout on, the optimizer stops checking for further query execution plans when it believes that the best plan it has found takes less time to execute than the amount of time already spent searching for a plan. If you issue a SET JOINOP NOTIMEOUT statement, the optimizer continues searching query plans. This option is often used with the SET QEP option to ensure that the optimizer is picking the best possible query plan.

Canceling Set Options

To cancel any of these options that have been set, you issue the opposite SET statement (SET NOOPTION to turn an option off or SET *option* in the case of JOINIP NOTIMEOUT to restore the default behavior).

The set session with on_error Option

The following SET statement enables you to specify how transaction errors are handled in the current session:

```
set session with on_error = rollback
statement| transaction
```

To direct Ingres to roll back the effects of the entire current transaction if an error occurs, specify ROLLBACK TRANSACTION. To direct Ingres to roll back only the current statement (the default), specify ROLLBACK STATEMENT. To determine the current status of transaction error handling, issue the SELECT DBMSINFO('ON_ERROR_STATE') statement.

Specifying ROLLBACK TRANSACTION reduces logging overhead, and help performance; the performance gain is offset by the fact that, if an error occurs, the entire transaction is rolled back, not the single statement that caused the error.

The following errors always roll back the current transaction, regardless of the current transaction error-handling setting:

- Deadlock

- Forced abort

- Lock quota exceeded

To determine if a transaction was aborted as the result of a database statement error, issue the statement SELECT DBMSINFO('ON_ERROR_STATE'). If the error aborted the transaction, this statement returns 0, indicating that the application is currently not in a transaction.

You cannot issue the SET SESSION WITH ON_ERROR statement from within a database procedure or multi-statement transaction.

I/O Tracing

SET IO_TRACE prints out information about disk I/O during the life of each query.

The following command turns I/O trace on as the default ING_SET level:

Windows:

```
SET ING_SET=set io_trace
```

UNIX:

C Shell:

```
setenv ING_SET "set io_trace"
```

Bourne Shell:

```
ING_SET = "set io_trace"
export ING_SET
```

VMS:

```
DEFINE ING_SET "SET IO_TRACE"
```

For example, given the query:

```
select * from iirelation;
```

Here is a sample output from the I/O trace. The counts are the number of pages read/written:

```
***************************************************************
I/O READ File: aaaaaaac.t00 (iidbdb, iirel_idx, 13) count:1
***************************************************************
***************************************************************
I/O READ File: aaaaaaab.t00 (iidbdb, iirelation, 0) count:8
***************************************************************
***************************************************************
I/O READ File: aaaaaaab.t00 (iidbdb, iirelation, 8) count:8
***************************************************************
I/O READ File: aaaaaaab.t00 (iidbdb, iirelation, 16) count:7
```

Note: When tracing the I/O or the locks of a parallel query (using SET IO_TRACE or SET LOCK_TRACE with SET PARALLEL *n*), the trace messages from child threads of the QEP are logged to the II_DBMS_LOG. The trace messages for the main thread are sent to the user session in the normal manner.

Lock Tracing

SET LOCK_TRACE prints out information about locks within a transaction.

The following command turns lock tracing on as the default ING_SET level:

Windows:

```
set ING_SET=set lock_trace
```

UNIX:

C Shell:

setenv ING_SET "set lock_trace"

Bourne Shell:

```
ING_SET = "set lock_trace"export ING_SET
```

VMS:

```
DEFINE ING_SET "SET LOCK_TRACE"
```

Event Tracing

The following features enable your application to display and trace events:

- To enable or disable the display of event trace information for an application when it raises an event, use the following command:

  ```
  set [no]printdbevents
  ```

  To enable the display of events as they are raised by the application, specify SET PRINTDBEVENTS. To disable the display of events, specify SET NOPRINTDBEVENTS.

- To enable or disable the logging of raised events to the installation log file, use the following statement:

  ```
  set [no]logdbevents
  ```

  To enable the logging of events as they are raised by the application, specify SET LOGDBEVENTS. To disable the logging of events, specify SET NOLOGDBEVENTS.

- To enable or disable the display of events as they are received by an application, use the following statement:

  ```
  set_sql(dbeventdisplay = 1 | 0)
  ```

  Specify a value of 1 to enable the display of received events, or 0 to disable the display of received events. You can also enable this feature using II_EMBED_SET.

You can create a routine that traps all events returned to an embedded SQL application. To enable/disable an event-handling routine or function, your embedded SQL application must issue the following SET_SQL statement:

```
exec sql set_sql(dbeventhandler = event_routine | 0)
```

To trap events to your event-handling routine, specify *event_routine* as a pointer to your error-handling function. For information about specifying pointers to functions, see your host language companion guide. In addition to issuing the SET_SQL statement, you must create the event-handling routine, declare it, and link it with your application.

User-Server Communications

The General Communication Facility (GCF) is composed of the GCA protocol and three communications programs: Name Server, Communications Server, and Data Access Server. These are separate processes. You can listen in on the communications between these GCF programs using the following Ingres environment variables and logicals:

- II_EMBED_SET "printgca"

- II_GCA_TRACE

- II_GCN_TRACE

- II_GCC_TRACE

- II_GCD_TRACE

- II_GC_TRACE

- II_GCCCL_TRACE

- II_GCA_LOG

Trace Communication Using printgca

You can trace communications occurring in the GCA with the following command:

Windows:

```
set II_EMBED_SET=printgca
```

UNIX:

C Shell:

```
setenv II_EMBED_SET "printgca"
```

Bourne Shell:

```
II_EMBED_SET = "printgca"
export II_EMBED_SET
```

VMS:

```
DEFINE II_EMBED_SET "PRINTGCA"
```

This traces all GCA messages passed between Ingres tools and the server. The output is placed into a file called iiprtgca.log.

Output of GCA Trace

Here is a sample output:

```
printgca = on session -2 (Thu Mar 26 15:50:32 2001)

GCA Service GCA_SEND
 gca_association_id: 0
 gca_message_type: GCA_QUERY
 gca_data_length: 49
 gca_end_of_data: TRUE
 gca_data_area: GCA_Q_DATA
 gca_language_id: 2
 gca_query_modifier: 0x01
 gca_qddata [0]:
 gca_type: 51: DB_QTXT_TYPE
 gca_precision: 0
 gca_1_value: 29
 gca_value: select name, gid from iiuser

End of GCA Message

GCA Service: GCA_RECEIVE
 gca_association_id: 0
 gca_message_type: GCA_TDESCR
 gca_data_length: 55
 gca_end_of_data: TRUE
 gca_data_area: GCA_TD_DATA
 gca_tsize: 26
 gca_result_modifier: 0x01
 gca_id_tdscr: 8
 gca_1_col_desc: 2
 gca_col_desc[0]:
 gca_attdbv:
 db_data: 0x00


 db_length: 24
 db_datatype: 20: DB_CHA_TYPE
 db_prec: 0
 gca_1_attname: 4
 gca_attname: (4) name
 gca_col_desc[1]
 gca_attdbv:
 db_data: 0x00
 db_length: 2
 db_datatype: 30: DB_INT_TYPE
 db_prec: 0
 gca_1_attname: 3
 gca_attname: (3) gid


End of GCA Message
```

```
GCA Service: GCA_RECEIVE
 gca_association_id: 0
 gca_message_type: GCA_TUPLES
 gca_data_length: 338
 gca_end_of_data: TRUE
 gca_data_area: GCA_TU_DATA not traced

End of GCA Message
```

Example: Level 2 Tracing Using GCA Trace Utiltiy

The syntax of GCA trace utility are as follows:

```
define II_GCA_TRACE [n]
define II_GCA_LOG [filename]
```

II_GCx_TRACE defines the level of GCA tracing, with n = 1 (lowest) through 4 (most detailed). II_GCA_LOG names the destination file of the trace output. If not specified, it defaults to standard output.

Here is an example of level 2 tracing:

```
!GCcm: target_id: 9062
!GCrequest 0: connecting on 9062
!GC_exchange 0: status 00000000
!GC_exchange_sm 0: status 00000000 state 0
!GCsend 0: send 528
!GC_send_comp 0: sent 528 status 00000000
!GC_exchange_sm 0: status 00000000 state 1
!GCreceive 0: want 528
!GC_recv_colmp 0: status 000000 state 0
!GCsend 0: send 65
!GC_send_comp 0: sent 65 status 00000000
!GCreceive 0: want 24
```

UNIX Trace Facilities

The UNIX operating system has various trace facilities that are useful in troubleshooting Ingres.

Bourne Shell -x Option

Invoking a Bourne shell with the -x option causes each shell command to be printed before it is executed. This is especially useful for diagnosing problems with installation shell scripts.

To use this trace on the Ingres shutdown script, call the Bourne shell (or Korn shell) with the -x option, passing the name ingstop as the script to be executed:

```
sh -x ingstop
```

This option can only be used with UNIX shell scripts, not with binary executables.

If you are in doubt as to whether a program or command is a binary or shell script, type:

```
file program_name
```

UNIX replies with "commands text," "ASCII text," or "executable shell script" if it is an executable shell script. Otherwise the file command returns a message specific to your machine—something like demand paged executable and possibly the designated chip on which it was compiled to run.

UNIX Trace Command

Some UNIX platforms (such as Sun) provide a facility to trace system calls as they are made by a process. To invoke it, enter trace at the operating system prompt.

Create an Ingres Console for Error Monitoring in UNIX

To ensure that any new errors written to the Ingres error log file are displayed on the terminal screen, issue the following command at the terminal.

```
tail -f $II_SYSTEM/ingres/files/errlog.log
```

To continuously monitor the activities of the Ingres recovery process (dmfrcp), issue this command:

```
tail -f $II_SYSTEM/ingres/files/iircp.log
```

Chapter 8: Managing Your System and Monitoring Performance

This section contains the following topics:

This chapter discusses the visual tools used to manage and monitor Ingres performance.

Note: The visual tools are available in Windows environments only.

Managing the Installation with Visual Manager

Ingres Visual Manager (IVM) provides a global view into the Ingres installation. It serves as a system console from which you can manage Ingres components and access other utilities. This utility captures events that are occurring in the system and allows them to be filtered for emphasis, based on your preferences.

IVM allows you to monitor (and start and stop) the different servers in the installation (DBMS, Name, Communications, JDBC, Data Access, Bridge, Star, and Recovery) as well as Remote Command, the Logging and Locking systems, and the Primary Transaction Log and Archiver process. It shows events occurring in the system both at the installation level and the level of each server. Using IVM, you can also edit the Ingres system and user environment variable settings.

Functions of Visual Manager

Ingres Visual Manager enables you to:

- View the started and stopped components and servers in the installation as well as the history of start/stop events

- Start or stop the entire installation or individual components

- View and edit Ingres system and user environment variables

- Control how various types of Ingres error log messages are handled

- Set preferences for event monitoring and notification

- View and filter events that occurred both at the installation level and the component or server level

- View statistics on these events

- View explanations for error log messages

- Access all Ingres visual tools

Visual Manager Window

The Ingres Visual Manager window contains a list of Ingres components. By selecting any branch of the corresponding tree, you can access information related to the branch, which correspond either to the entire Ingres installation (the root branch), an individual component configuration, or an individual instance of a given configuration.
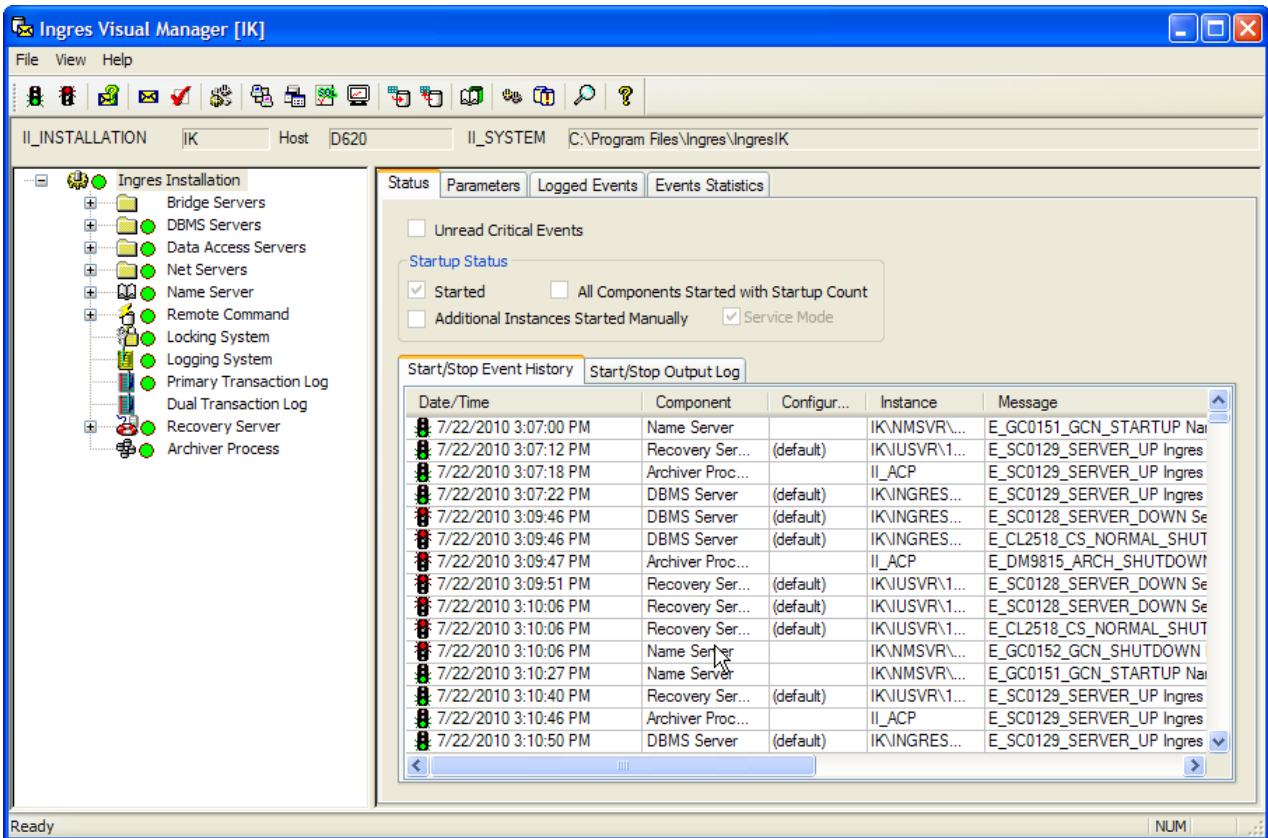
The Visual Manager Window contains the following tabs:

Status

Displays the summary of startup status of the components of the installation, the start and stop history of any component in the configuration, and related information such as the output of the last start or stop operation.

Parameters

Displays system, user, or extra parameters.

Logged Events

Displays a list of logged events for the Ingres installation and any of its sub-branches. You can also select an event to view its corresponding message text.

Event Statistics

Displays statistics on events that have occurred for the selected component, grouped per event ID, and presented as a numerical count and in graph form.

For more information about these pages, see online help for Ingres Visual Manager.

System and User Parameter Configuration Through IVM

The Parameters page under the Ingres Installation branch contains lists of parameters for system, user, and extra parameters along with their values and descriptions. You can add or unset a parameter and edit its value. You can also choose whether you want to view Ingres parameters that are not currently set.

For more information, see the online help topic, Parameters Page, Ingres Installation branch (Ingres Visual Manager window).

Set Parameter Configuration Through IVM

From IVM, you can access the Configuration Manager utility to configure parameters for each server component in the Ingres installation. To access Configuration Manager, select the desired server component in the left pane of the Ingres Visual Manager window and click the Configuration Manager toolbar button, or choose File, Configure.

For more information, see the online help topic, Configuring Ingres Components in the Ingres Visual Manager online help.

Message and Notification Management Through IVM

Using IVM, you can control how various types of Ingres error log messages are handled. The Define Message Categories and Notification Levels dialog lets you specify which messages in the errlog.log must be discarded, which are displayed normally in IVM, and which trigger a special alert. This dialog also allows you to group messages according to user-defined categories, which can be handled as a group for discard, display or alert.

To access the Define Message Categories and Notification Levels dialog, click the Categories and Notification Levels toolbar button, or choose File, Categories and Notification Levels. For information on using this dialog, see the Ingres Visual Manager online help.

Event Monitoring Management Through IVM

Using the Preferences dialog in IVM, you can specify the following settings related to event monitoring.

- Maximum amount of memory used to store events (in megabytes)

- Maximum number of events stored

- Maximum number of days that the event is stored before being deleted

- Only messages that arrived after the last Name Server startup are stored

To access the Preferences dialog, click the Preferences toolbar button, or choose File, Preferences.

Alert Events Notification Settings in IVM

When an Alert message is written to the Ingres errlog.log file, IVM indicates the alert through a special icon change in the tray toolbar and in the IVM window tree. IVM also lets you set additional preferences for alert notification using the Preferences dialog. These additional preferences include:

Sound

When selected, you are alerted of an event by a sound (beep).

Message Box

When selected, you are alerted of an event by a message box if there are unread "Alert" messages.

OS Event (for new Events only)

When selected, the full text of all "Alert" messages resulting from new events are logged in the Operating System [Application] Event log in addition to the Ingres errlog.log file. If this option is selected, you can also set preferences for generating specific Operating System events.

To access the Preferences dialog, click the Preferences toolbar button, or choose File, Preferences. For instructions on setting these additional preferences, see the Preferences dialog topic in the Ingres Visual Manager online help.

View Message Explanations in IVM

IVM allows you to view an explanation for any Ingres message in the errlog.log file. To view an explanation for a message in the errlog.log file:

1. Select the desired message on any page or window in IVM that allows you to select messages.

2. View the corresponding explanation by clicking the Message Explanation toolbar button.

For details about the Message Explanation window, see the Ingres Visual Manager online help.

Component Monitoring Through IVM

You can monitor the following components through IVM

- Servers

- Logging System

- Locking System

Server Monitoring Through IVM

For each of the following server components as well as their instances, you can use Ingres Visual Manager to monitor status, logged events, and event statistics:

- Bridge Servers
- DBMS Servers
- Data Access Servers
- Communications Servers
- Star Servers
- Name Servers
- Remote Command
- Recovery Server—a Recovery Log File page appears in the IVM window
- Archiver Process—an Archiver Log File page appears in the IVM window

For more information on monitoring server components, see online help for Ingres Visual Manager.

Logging System Monitoring Through IVM

The Logged Events page displays a list of the logged events for the Logging System branch, and any of its sub-branches. However, the messages are filtered according to certain preferences you define in the Preferences dialog.

For more information, see online help for Ingres Visual Manager.

Locking System Monitoring Through IVM

The Logged Events page displays a list of the logged events for the Locking System. However, the messages are filtered according to certain preferences you define in the Preferences dialog. For more information, see online help for Ingres Visual Manager.

Access to Visual Tools Through IVM

IVM provides direct access to all Ingres visual tools. (The only exception is the Ingres Service Manager, whose functionality is already provided by IVM.)

You can access Ingres tools in the following ways:

- Right-click the tray toolbar to display a menu of visual tools.

- Select a tool from the IVM toolbar:
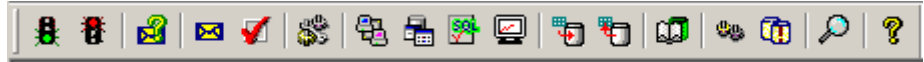
Note: Only the most frequently used tools are accessible from this toolbar.

Monitoring and Managing Ingres with Visual Performance Monitor

Visual Performance Monitor can be used as a monitoring tool, a performance analysis tool, and a system management tool.

Visual Performance Monitor lets you monitor the following:

- Servers

  You can view a list of the servers that are started on an Ingres installation, and information about each server. For example, you can view the sessions that are currently active for each server. You can also remove a session or stop a server (if you are a privileged user).

  Visual Performance Monitor provides the following information for a particular server:

  - Lock lists

  - Locks

  - Transactions

  - Locked databases, tables, and pages

  - Other locked resources

  - Network traffic for Net servers

- Users and sessions

  You can monitor users for whom there are open sessions. You can find out which sessions are open for a particular user, and drill down further to reveal all the related information about those sessions (lock lists, transactions locked, databases locked, and so on).

- Logging

  You can view logging system summaries, transaction lists, process, and database lists. Log information can be used to monitor transaction rates, log file activity, processes, and databases in the logging system. This information is useful in determining which logging parameters need to be adjusted.

- Locking

  You can monitor lock information to help determine which lock parameters need to be adjusted. Viewing locking system summaries, lock lists, and resources provides you with the information you need to spot conditions when, for example, additional locking system resources need to be added.

  Viewing your lock lists is useful for locating transactions that cannot proceed because they are blocked by another transaction.

- System performance

  You can view your performance information from a "database" point of view. This means you can access performance information using a database branch, as opposed to the root branches of Visual Performance Monitor.

- Replication

  You can start, stop, and monitor Replicator servers that are required for the replication scheme that has been defined in a VDBA DOM window. You can set up startup parameters for these servers, send events to these servers, view and manage collisions, and display other miscellaneous replication monitor information.

Visual Performance Monitor lets you perform the following actions:

- Refresh data in the window from a server
- Shut down a database server or close a session

Visual Performance Monitor Window

The Visual Performance Monitor window contains a list of categories under which various types of Ingres performance information appears. Under each root object category are branches representing each performance entity. Under each branch are one or more sub-branches that pertain to the particular type of performance entity.
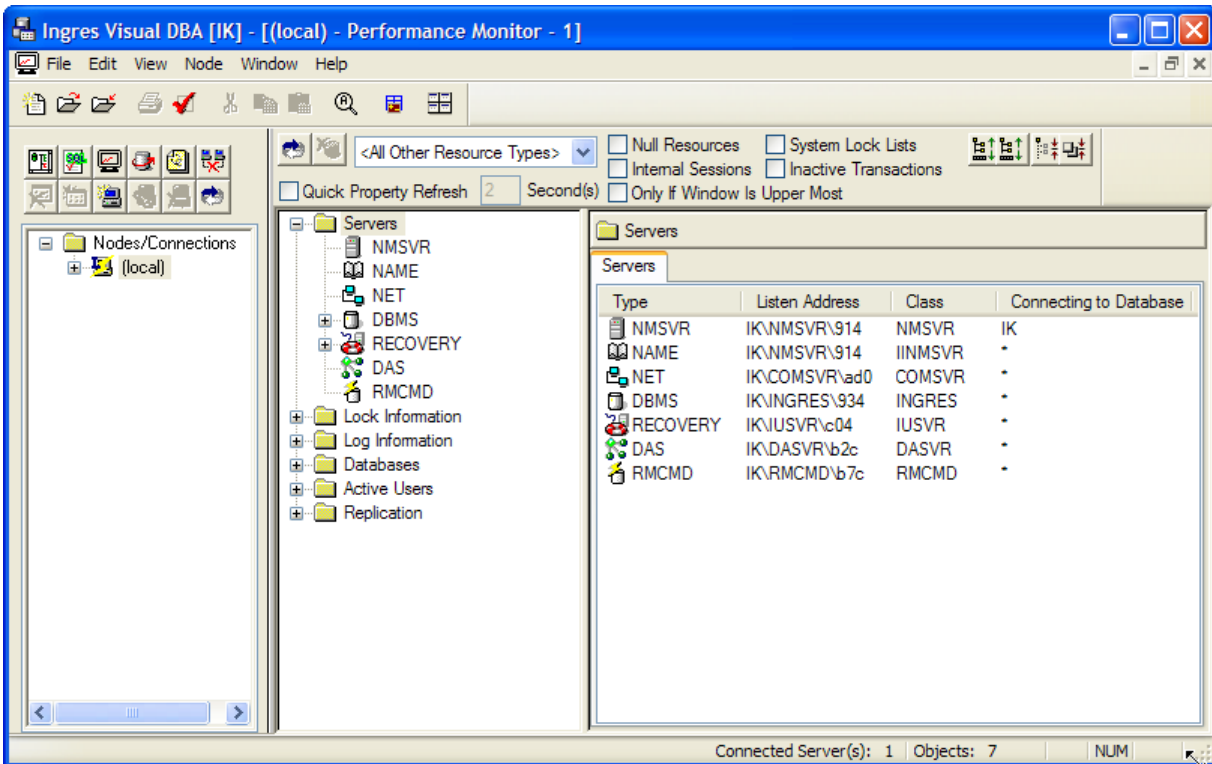
For more information, see the Visual Performance Monitor online help. For context-sensitive help on any active dialog or window, press F1.

Comparing Configurations with Visual Configuration Differences Analyzer

The Visual Configuration Differences Analyzer (VCDA) allows you to compare the configurations for an Ingres local installation by taking "snapshots" of its configuration. For example, after you install Ingres and have set up the database schemas, take a snapshot of the configuration. If you encounter problems later on, take another snapshot of the configuration and compare it to the earlier snapshot to see if any configuration changes have contributed to the problem. Keep an on-going record of configuration changes by starting with a snapshot of a new installation that shows all system defaults, and taking a snapshot of the installation each time the configuration is changed.

Note: VCDA compares the configurations of Ingres installations, but not the database object definitions within these installations. To compare database objects, see the Visual Database Objects Differences Analyzer (VDDA), in the *Database Administrator Guide* and the Ingres online help.

Specifically, VCDA allows you to:

- Save a snapshot of the current configuration into a file
- Compare two saved snapshot files, or the current installation with a saved snapshot
- Restore selective groups of configuration parameters from a saved snapshot

For specific instructions on performing these tasks, refer to the VCDA online help.

Configuration Snapshot File

When you take a snapshot of the current configuration for the local installation, VCDA writes the following information to a text file:

- Main Parameters: The value of II_SYSTEM, II_CLUSTER, II_CONFIG_LOCAL, II_NETBIOS_NAME, hostname, and username
- All information included in the config.dat file, and in the additional config.dat files pointed to by II_CONFIG_LOCAL (if any)
- All vnode information, except for password data
- All Ingres system variables
- All Ingres user variables

This snapshot can be compared to other snapshots as explained below.

Comparison of Configuration Snapshots

VCDA allows you to compare the current configuration of the local installation with a previously saved configuration snapshot. It also allows you to compare two saved snapshots that do not include the current configuration. VCDA also indicates, using an icon color, if the difference is due to a parameter that is different in both snapshots, or a parameter that exists in one snapshot but not in the other.

If there are no differences in the main parameters, VCDA displays a green icon and a message. If there are no differences in the other information, VCDA displays only a green icon.

Comparison of Remote Installation Snapshots

Although you can only take snapshots of the local installation, you can compare configurations on remote installations (even on different platforms) by making the snapshot file from one machine available to another machine that is running VCDA. The snapshot file is a simple text file that is readable across different platforms.

Restoration of Configuration Parameters Through VCDA

In addition to comparing configuration snapshots, VCDA allows you to restore selected groups of configuration parameters from a saved snapshot. The only parameters that VCDA does not restore are the values of II_SYSTEM, the hostname and username.

You restore parameters by first selecting the snapshot file that contains the desired parameters. You can specify the specific parameters you want to restore. If the snapshot file you choose corresponds to a networked installation, you can also specify the value of the corresponding Ingres variables and the content of the relevant additional config.dat file, with an option to create the tree structure if it does not exist.

VCDA automatically creates a backup snapshot file of the current configuration before it restores parameters. If for some reason you do not want to keep the restored parameters, you can select this backup snapshot file to restore the parameters to their previous state.

How VCDA Handles Concatenated config.dat Files

VCDA uses information from the config.dat file, which normally contains parameters that apply to the local hostname only. However, if you want to concatenate the contents of all your config.dat files into a single config.dat file to distribute across multiple environments, VCDA manages this situation as follows:

■ The hostname under which a snapshot is saved shall be saved within the snapshot.

■ If VCDA detects that hostnames other than the snapshot one are managed within the config.dat information of the snapshot, VCDA shall display it.

A "hostname mapping" option is available in that situation, so that VCDA can compare the additional hostnames configuration parameters. If this option is not used, the parameters are compared including their "hostname" part, (that is, only parameters that are identical for the same hostname within the two snapshots are considered as identical).

Chapter 9: Analyzing and Recovering Journaled Transactions

This section contains the following topics:

This chapter discusses journaled transactions and how to analyze, recover, and redo them using the Journal Analyzer visual tool.

Journal Analyzer

The *Journal Analyzer* is a utility that allows you to view and analyze journaled transactions and individual underlying statements. It also enables you to create SQL scripts, which are used to recover or redo individual row operations without redoing or rolling back the whole database or table.

The Journal Analyzer is supported in Ingres 2.6 and above.

In the Journal Analyzer, you can:

- View past transactions and their begin/end timestamps, duration, LSN, and user, and the number of insert, update, and delete statements within the transaction

- View tables and rows involved in the transaction

- View before and after images and corresponding LSN for row changes within the transaction

- View rolled back transactions

- View some types of statements, such as create table or create index, inserts, updates, and deletes, and recover or redo journaled insert, update, and delete statements

- Recover and redo transactions immediately or generate scripts to do it at another time or in another database

- Recover and redo multiple transactions at one time, allowing, for example, the replaying of whole sequences of journal files on a remote node/database

- Recover and redo individual row changes

For detailed instructions on using the Journal Analyzer, see the Journal Analyzer online help.

Start Journal Analyzer

To start the Journal Analyzer:

Windows:

Do **one** of the following:

- Click Start on the taskbar and choose Programs, Ingres, Ingres Journal Analyzer.

- From within Ingres Visual Manager's right-click popup menu, select Journal Analyzer, or click the Journal Analyzer toolbar button.

- Type **ija** at the command prompt.

Journal Analyzer Window

In the Journal Analyzer window, the left pane consists of the vnode, databases, and tables. In the left pane, you can drill down to the desired vnode, database, and (optionally) table. When a database or table has been selected, the right pane appears as follows:
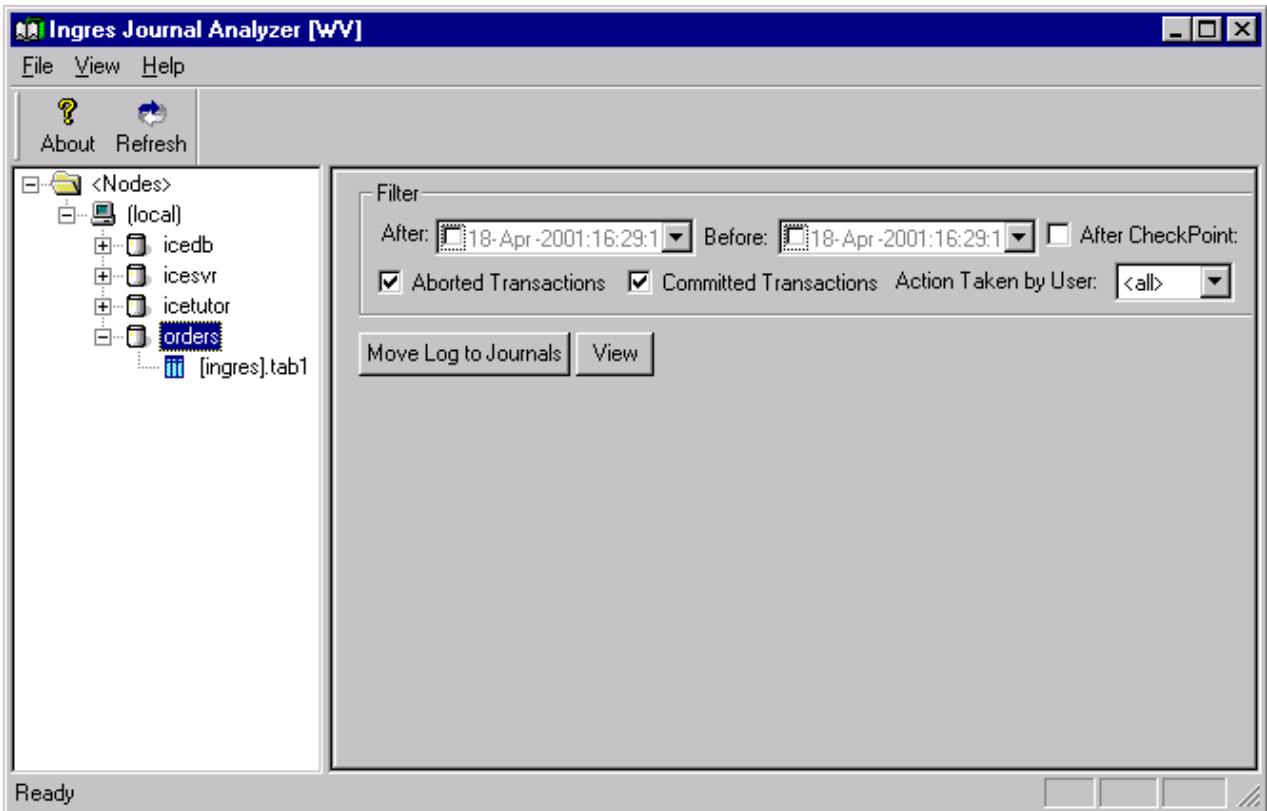
Controls in the right pane are as follows:

Filter

Use the Filter options to control the quantity and type of transactions that are displayed, based on different criteria such as the transaction date and time, checkpoint number, user, and so on.

View

Clicking the View button displays transaction information that is currently in the journals. You must click the View button to refresh the data when you change the filter criteria.

Move Logs to Journals

To see any recent transactions that are in the transaction log file—but have not yet moved to the journals—click Move Log to Journals.

Transaction Views

Based on whether you select a database or table in the navigational pane of the Journal Analyzer, and whether the Group by Transaction option is selected, you see a slightly different interface in the right pane.

Database Level Journaled Transactions

When you select a database, you see the journaled transactions within the database that match the criteria defined in the top of the pane. You can then determine whether you want to group transactions and view the details for only selected transactions.

Group Changes by Transaction

When the Group by Transaction option is selected, a single line of "summary" information is displayed for each transaction in the uppermost list, sorted by transaction number. When one or more of these lines are selected, the detailed information for all selected lines is shown in the lower list. The detail information corresponds to each row of journaled tables altered in the transaction(s), in addition to statements such as create table or alter table that were executed within the transaction.

For example, in the following window, a transaction is selected and the corresponding detail lines are displayed beneath it:
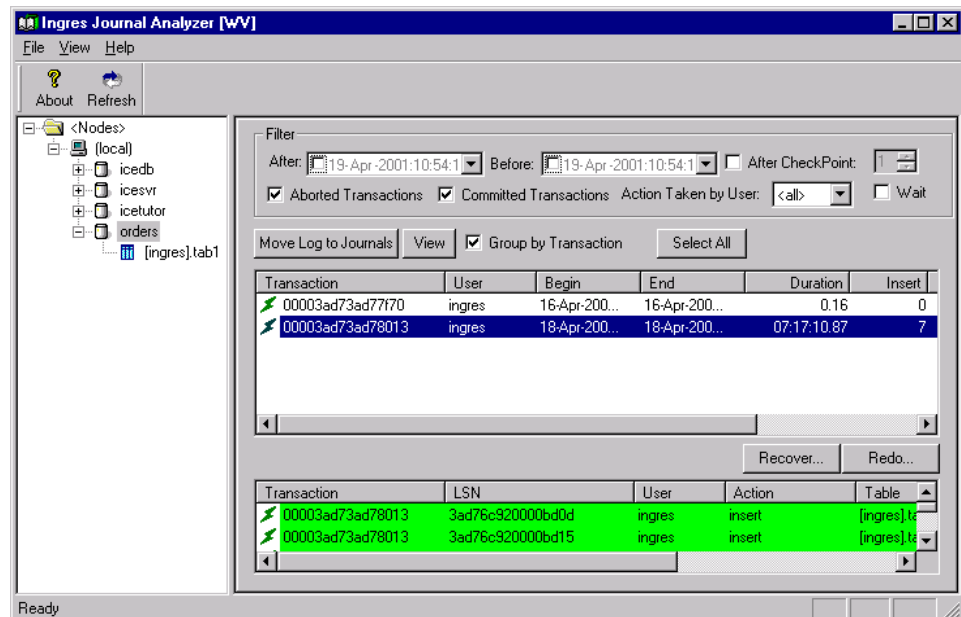
Although the rows are sorted by transaction number initially, look at (and sort by) transaction end (End column), because it is the time when all changes for the transaction become available to other users. A transaction with a higher number (that is, having started later) than another transaction, can be committed first, in which case the underlying changes become available to other users before those of the other transaction (which had a smaller number because it had started first). For this reason, sort on transaction end (by clicking on the corresponding header).

View Transaction Details

If the Group by Transaction option is not selected, a single list appears, displaying all the individual actions for each transaction, in transaction order:
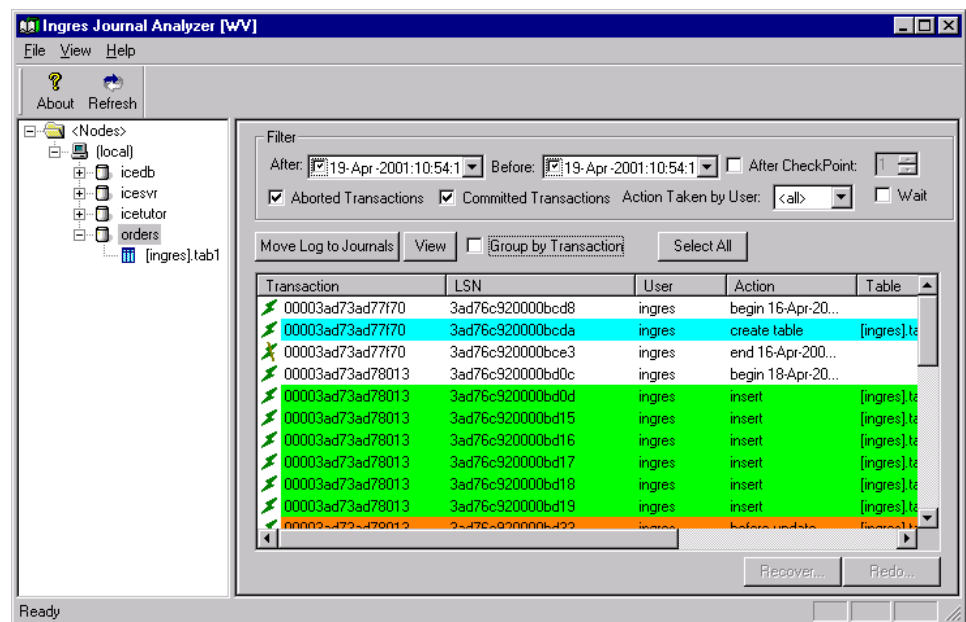
Even these detail lines provide a summary of information on the individual actions. This list cannot display column names because different tables have different column names. To get the full detail of individual rows that have changed, you must choose a table in the navigational pane. See Table Level Journaled Transactions (see page 232).

For update statements, the "before update" and "after update" images are shown. For inserts and deletes, only one row (the inserted or deleted one) appears. Different colored rows are used to denote the type of statement (in addition to the Action column that displays strings such as begin, end, insert, update, delete, and so on).

An icon preceding each line shows you whether the transaction it belongs to was committed or rolled backed, included a partial rollback, or involved changes in non-journaled tables (an exclamation point next to the icon appears in this case).
You can sort the data by any criteria by simply clicking on the corresponding column header.

Table Level Journaled Transactions

On the table level, the right pane looks and functions similar to the one at the database level, but only the row changes corresponding to the selected table appear. There is no Group by Transaction check box for the table level. At the table level, you can view a history of journaled changes in that table.
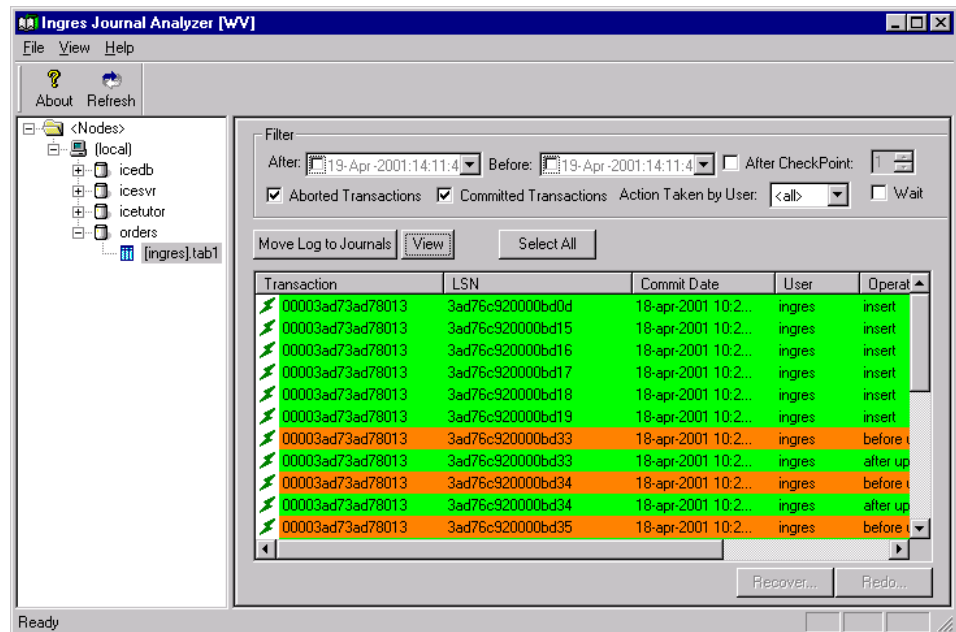
As opposed to the database level panes, where more than one table was involved, it is possible here to have the real column names become headers of the results list, which was not possible at the database level because the columns are normally different in each table that was involved in the transaction.

You can also sort the data on the table level by clicking on a column header. In particular, the following sorts are useful:

■   Commit date—this is the most meaningful criteria for sorting the changes

■   Transaction number—be aware that the transactions are not always committed in the same order

■   LSN—shows you the exact sequence of row changes

For more information on the interface for this utility, see the Journal Analyzer online help topics.

Select Transactions and Row Changes

After browsing for the transactions or row changes that you want to recover or redo, you can select one or more transactions or individual row changes. If the Group by Transaction check box is selected, and multiple transaction lines are selected, the detail list shows the row changes corresponding to all selected transactions. If this option is not selected, you are selecting lines at the table level, which correspond to row changes, not full transactions. The Recover and Redo dialogs proposes (but does not pre-select) the other changes in those transactions of the initial row changes that were selected.

Tip: Select whole transactions associated with the changes to be recovered to ensure transaction integrity. This is the preferred and most common way to recover row changes.

After making your selections, the Recover and Redo buttons are available. If you do select lines corresponding to only *part* of a transaction, there are other factors to consider, as described in Recover Transactions or Individual Row Changes (see page 234) and Redo Transactions and Individual Row Changes Option (see page 241).

The Select All button allows you to select all transactions currently displayed in the pane for the selected criteria. This can be useful for example for replaying whole portions of journals on another node/database, through the Redo functionality described in the Redo Transactions or Individual Row Changes (see page 241).

Recover Transactions or Individual Row Changes

The Recover dialog is used to create one of two types of "undo scripts"—an internal script that is run immediately or an external script that can be saved and run later.

To recover transactions, follow these steps:

1. When you click Recover, the Recover dialog appears as follows:
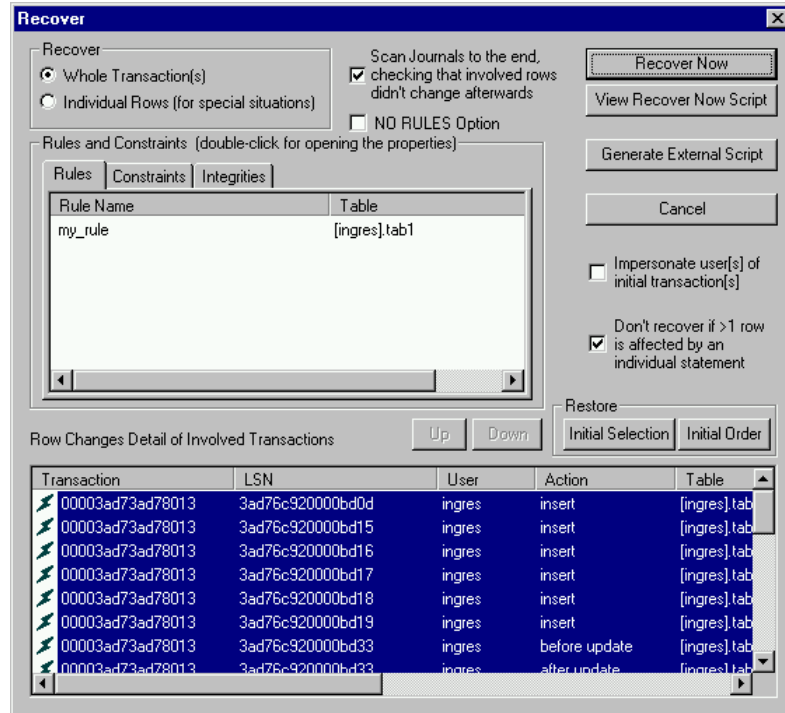
Initially, the row changes that you selected in the Journal Analyzer are displayed in the lower portion of the Recover dialog.

2. Click "Recover Now" to recover immediately, or "Generate External Scripts" to generate scripts that can be run later, as described in Recover Immediately or Generate External Scripts Option (see page 239).

Recover "Whole" Transactions or Individual Rows

If you chose whole transactions in the Journal Analyzer, the Whole Transaction(s) option is selected automatically in the Recover dialog.

If you selected individual row changes that make up only part of a transaction (in the lower portion of the dialog), the Individual Rows option is selected. In this case, you are prompted for confirmation as to whether you want to recover only parts of transactions, and lose transaction integrity. Although the recovery of individual rows is not recommended, it can be useful, for example, if a table has Ingres rules associated with it.

Rules, Constraints, and Integrities

If a table has Ingres rules associated with it that results in additional row changes within the same transaction, do not generate undo operations for such dependent changes because it leads to undesirable results. Also, the rules can change between the time of the initial transaction and the time when you want to undo that transaction. Instead, undo only the change in the base table, and let the rules generate the dependent changes.

Journal Analyzer detects the rules, constraints, and integrities that exist in the tables involved in the transaction(s) and displays them on the tabbed pages in the Recover dialog. From this display, you can determine whether some changes in a transaction resulted from rules and if certain constraints cannot be fulfilled. This information helps you determine if you want to use the No Rules option, and possibly disable certain rows within the transaction. You can view the properties for a particular rule, constraint, or integrity by double clicking it. Examine these entities to determine their impact on a restore.

Note: Integrity constraints that are violated are not specifically reported: updates and inserts that violate any integrity constraints are simply not performed.

Scan Journals to End Option

The Scan Journals to the End option provides a warning if the tables have been altered since the time of the initial transaction, in which case the rules for these tables have changed.

If there are rules on a table, you have two choices:

- Disable individual row changes in the undo scripts, by deselecting the corresponding row changes in the Recover dialog. This can be useful, for example, if the current rules cause the desired operation in dependent tables, in which case you must disable the dependent row changes.

- Use the No Rules option to allow you to disable rules while executing the queries. This option allows the reversal of the exact row changes of the initial transaction, regardless of what the rules were at that time, and what they are now. The side effect of this option is that if there are constraints on the corresponding tables, *they are* not *checked if this option is used*. This operation must be used very carefully. If the Journal Analyzer previously detected current constraints on those tables, it provides a warning and asks for confirmation before starting the operation.

Order of Transactions and Statements in Transactions

The order of the undoing of transactions is reverse of the order at commit time of these transactions and cannot be changed. If for some reason you need the order to be different than the usual order, the only way to do this is to recover each transaction separately (by launching the Recover dialog for each transaction individually) in the desired order.

In each transaction to be recovered, the order of the individual statements can be changed, using the Up and Down buttons in the Recover dialog.

How to Check If Rows Have Changed After the Transaction

Before proposing to reverse row changes resulting from a transaction, you must take into account the fact that some of these rows have changed after the transaction. The possible cases are as follows:

- None of these rows have changed and there is no problem.

Or:

- If a row has changed, usually no row corresponding to the "after image" of the initial change can be found, and therefore the "reverse" statement, which includes a where clause that searches for a row corresponding to such "after image," fails because no such row is found.

  There is an "advanced" exception, in the case where a given row (the same or another one) has the exact same content afterwards as the after image of the initial change: in this case, it is the user's responsibility to make a decision about taking additional actions.

For this reason, the following features are available:

- The changes that you have asked to reverse are displayed in the dialog for you to view.

- By default, the Scan Journals to the End option is selected. It indicates if later changes affected the rows involved in the transaction. If this problem is not an issue, disable the option.

- If rows of the transaction to be undone have changed since the transaction, Journal Analyzer prompts you with a warning, asking whether you want to continue.

Users of Transactions

By default, Journal Analyzer proposes to recover transactions by executing reverse transactions under the currently connected user. However, the Impersonate User of Initial Transaction option allows you to impersonate the user under which each transaction had initially been executed.

Note: This option requires that the connected user is the database administrator or has appropriate privileges.

The Impersonate User of Initial Transaction option affects the way the reverse transactions are executed: that is, normally Journal Analyzer recovers multiple transactions by executing the recover statements of all transactions to be undone in a single, global "recover" transaction. This allows, in case of failure at any point in the recover operation, to cancel the whole operation.

However, given that it is not possible to execute in the same transaction different statements under different users, if this option is used, and the initial transactions had not all be executed under the same user, Journal Analyzer must commit (and change the user) before each reverse statement to be executed under a different user than the previous one.

A possible problem in this case is when a reverse statement fails, once other recover statements have already been committed. In this situation, Journal Analyzer proposes to "undo the undo" of the transactions previously undone. However in case of failure of this last operation, you need to manually "repair" this situation (typically by using Journal Analyzer) by completing or canceling the uncompleted operation.

## Number of Rows Affected by Each Individual "Reverse Statement"

Journal Analyzer-generated reverse statements for insert and update statements are delete and update statements, with a "where" clause on all columns that can be accepted in a where clause (column types such as "long varchar" cannot).

If you need two identical rows in which all columns can be part of a where clause (SQL does not normally distinguished such rows), the Recover dialog has an option (selected by default) called "Don't Recover if >1 Row is Affected by an Individual Statement". Only when this option is explicitly disabled does the recover process work even if more than one row is affected by an individual recover statement.

In the case where no row is affected by an individual "recover" statement, Journal Analyzer provides an error anyway, and the recover operation fails. If the Scan Journals to End option has not been disabled, Journal Analyzer normally warns you that rows involved already have changed since the initial transaction (different warnings appear if journaling has been disabled after the initial transaction or in similar situations).

Recover Immediately or Generate External Scripts Option

If you want to view the script that is run based on your Recover dialog selections, click the View Recover Script Now button. The following dialog appears:
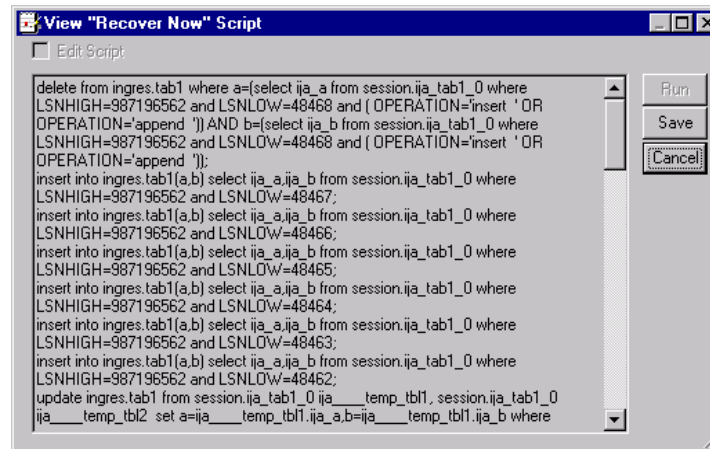
This dialog allows you to see the SQL statements that are executed if you click the Recover Now button. The script only works when executed within Journal Analyzer and must not be used externally. If desired, you can also save the script by clicking Save.

If you want to generate an external undo SQL script that you can run at another time (possibly on another database), you can do so after you have made your selections in the Recover dialog. By clicking the Generate External Script button, the following dialog appears:
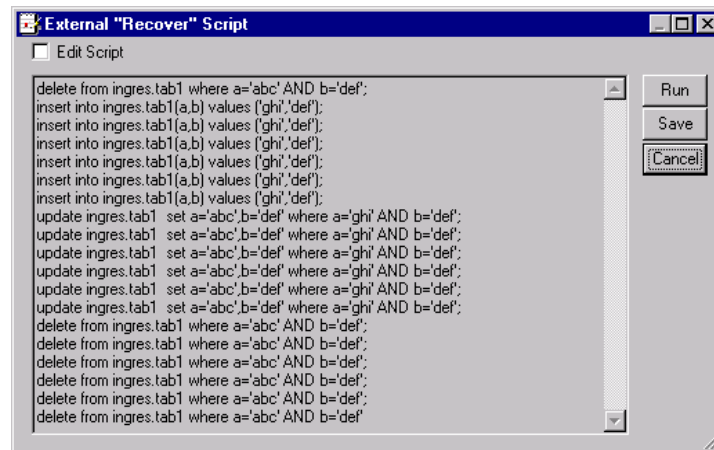
You can make any necessary changes to the script by selecting the Edit Script option. To save or run the script, clicking the corresponding button.

Although external scripts can be very useful, you should be aware of some issues:

- Binary data cannot be conveniently inserted into an SQL script

- Floating point numbers can be rounded if inserted into a script in text format

For this reason, the Generate External Script options must be restricted for the Recover and Redo features to when absolutely necessary (that is, typically when executing the scripts on another node and database). In this case:

- There is an error message if some data types included in rows of the transaction cannot be inserted in the script

- The user must be aware of the context of possible rules and constraints existing on the target database at the time when the script is executed and accordingly update the script.

Redo Transactions and Individual Row Changes

From the Journal Analyzer window, you can also launch the Redo dialog, as follows:
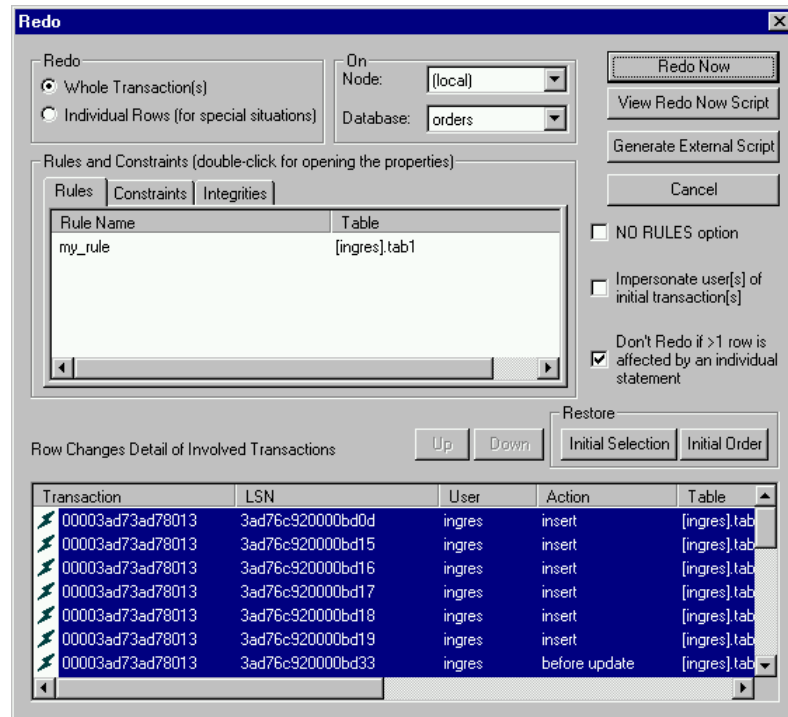
This dialog is useful for the following purposes:

- For redoing rolled back transactions (that had been rolled back for an unwanted reason for example)

- For regenerating similar, committed transactions in another database

This dialog is similar to the Recover dialog; however, two additional controls allow you to specify a different node and database where to redo the transaction or statement. All the features described in the Recovering Transactions section for the Recover dialog apply to this dialog and function in the same way—only for redoing transactions. For more information, see that section and the help system.

Chapter 10: Understanding Ingres Management Architecture

This section contains the following topics:

This chapter describes the features of the Ingres Management Architecture (IMA), restrictions, and examples of how to use it.

IMA Audience

IMA provides the means for developers and third-party vendors to create applications to manage and monitor Ingres installations.

These capabilities are of interest to:

- Database administrators

- System administrators

- Developers of tools for monitoring and managing Ingres

The user of IMA must be familiar with SQL-based tools such as OpenROAD, Ingres Terminal Monitor, Visual Basic, or ODBC and have an overall understanding of Ingres and of Ingres system monitoring and performance.

IMA Overview

IMA provides an SQL-based interface to a set of user-registered tables accessing management information to allow the monitoring and managing of an Ingres installation.

IMA exposes for monitoring data that is currently reported through Ingres utilities such as:

- Ingres Visual Manager

- Visual Performance Monitor

- Interactive Performance Monitor (ipm)

- Lockstat

- Logstat

- iimonitor

- Trace points

Through IMA, system management applications can be created that allow management operations to be performed against Ingres servers. These operations include:

- Stopping DBMS and Star servers

- Removing DBMS sessions

IMA-based management tools can monitor and control the local environment or an Ingres remote installation accessed through Ingres Net. By combining IMA with Ingres Net, local and multiple remote Ingres installations can be managed and monitored from a single point.

Components of IMA

The IMA architecture is represented in the following diagram:
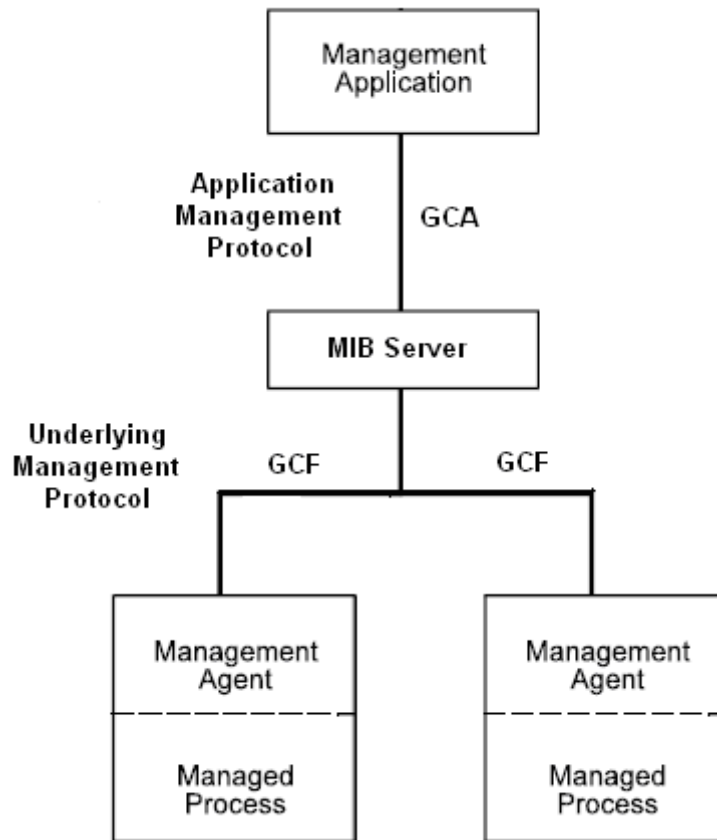
The user interfaces with an SQL-based **Management Application** program, for example, OpenROAD.

The Management Application interfaces with the **MIB Server** across the **Application Management Protocol**. Ingres has its own communications protocol that uses the **GCA** (General Communications Architecture).

GCA is the communications interface used by clients, servers, and distributed data managers to access GCF services that passed tuples, queries, and commands between Ingres processes. Through this mechanism, the Management Application receives MIB data from the MIB Server.

In the current IMA architecture, a MIB Server is a DBMS server.

The GCA allows the Management Application to either connect directly to a MIB Server or indirectly to a remote MIB Server using a GCC process.

The IMA components in the MIB Server are as follows:
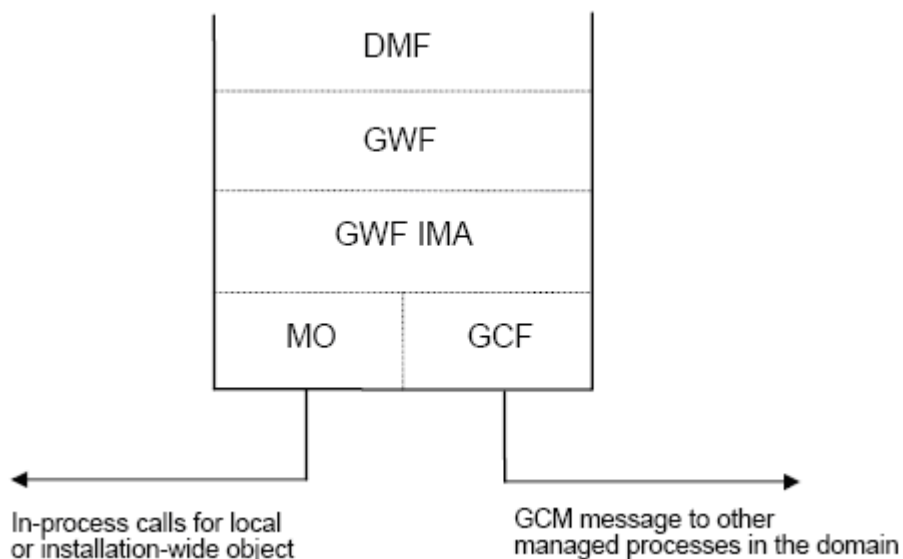
Component	Description
DMF	Data Management Facility
GWF	Gateway (Enterprise Access) Facility
IMA GWF	IMA Gateway Facility
MO	Managed Object module for handling local DBMS IMA objects
GCF	General Communication Facility for communicating with other Ingres processes
GCM	General Communication Mechanism for communicating with other Ingres processes

The MIB Server uses IMA GWF, a non-relational gateway to present the IMA objects as relational database tables.

Without a MIB Server, IMA data is not available to ordinary users.

For objects internal to the DBMS Server, the MIB Server reads and writes objects directly through MO. Objects in other processes, for example a communication server, are handled using the GCF message protocol.

A Managed Process only needs to talk a single protocol, while MIB servers can support Management Applications using a variety of protocols.

IMA has been designed to be compatible with other management protocols, for example:

- SNMP (Simple Network Management Protocol)
- DME (Distributed Management Environment)
- CMIP (Common Management Interface Protocol)

Management Information Base

The DBMS server acts as a MIB Server.

MIB means Management Information Base, which in SNMP terms is a set of information in any system that can be monitored and manipulated for the purposes of system management.

From the standpoint of the monitoring application, the MIB is simply a set of tables in a relational database that can be queried and modified using SQL statements. In actuality, the MIB is an abstraction—it is composed of internal information spread among all the Ingres processes in an installation, including:

- The DBMS Server
- The archiver
- The recovery process
- The Name Server
- Communication servers
- Gateways
- Slave processes

The MIB server provides a central point of contact for the monitoring application, and acts as a distributed gateway to the actual MIB data, which in its natural habitat is not relational and does not reside in the same location.

The individual units of data are often referred to as "MIB objects" or "MIB variables" and most often are program variables in the code that runs the managed process.

A MIB object has a two-part name:

The object class

A generic variable type

An object instance

A unique identifier constructed for each actual manifestation of a MIB variable

Each object class also has a set of permissions defined for it that is used by the MIB server to enforce access control. This prevents unauthorized users from "playing" with the system.

Permissions are octal values:

Permission	Octal Value	Description
MO_PERM_NONE	0000000	none
MO_SES_READ	0000002	read by session
MO_SES_WRITE	0000004	write by session
MO_DBA_READ	0000020	read by DBA
MO_DBA_WRITE	0000040	write by DBA
MO_SERVER_READ	0000200	read by server administrator
MO_SERVER_WRITE	0000400	write by server administrator
MO_SYSTEM_READ	0002000	read by system administrator
MO_SYSTEM_WRITE	0004000	write by system administrator
MO_SECURITY_READ	0020000	read by security officer
MO_SECURITY_WRITE	0040000	write by security officer
MO_ANY_READ	0200000	override read perm bits
MO_ANY_WRITE	0400000	override write perm bits

For example:

- The class ID "exp.gcf.gca.trace_level" has a permission value of 28086, which is a decimal value. The octal value is 66666, which is:

  MO_SES_READ + MO_SES_WRITE + MO_DBA_READ + MO_DBA_WRITE + MO_SERVER_READ + MO_SERVER_WRITE + MO_SYSTEM_READ + MO_SYSTEM_WRITE + MO_SECURITY_READ + MO_SECURITY_WRITE

- The class ID "exp.gwf.gwm.session.control.add_vnode" has a permission value of 6, which is a decimal value. The octal value is 6, which is:

  MO_SES_READ + MO_SES_WRITE

MIB Object Model

The MIB is defined as an abstract entity consisting of simple type objects. *Simple type* objects are objects that have values that are strings or strings that can be interpreted as integers. Simple type objects do not represent an aggregate such as a structure.

Aggregate objects, such as a C language structure, are represented in the MIB by having a separate object type for each element. This can be compared to having a single table for each column in a relational database, with the entire table made visible with a join of all the column tables.

In the model each object has:

- classid
- location
- one or more instances

Classid

Each data object in the MIB has a name, called the classid, which defines its type and semantics. A classid is a string that forms a hierarchical name space by using a period (.) to separate levels of the tree. For example:

```
exp.scf.scs.scb_query
```

In this example, the object is identified as an experimental object (exp). In the current release, all objects are under the experimental branch of the name space tree, owned by the session control facility (scf), in scs code block, and is the scb_query object.

Location

Each data object is located in a place, where the general format of a place string is:

[*vnode*::] [/@iigcn | /@*gca_address*]

where:

vnode

Is the IMA domain (also know as the Ingres installation)

gca_address

Is the listen address of the server as shown in Visual DBA, Interactive Performance Monitor (IPM), or Name Server Maintenance utility (iinamu).

If the place is visible in several different processes in an installation, then it is "vnode specific."

If the place is a particular server, then it is "process specific."

The place can be a local or remote installation. To connect to the remote installation, an installation password must have been defined.

Instances

A classid can have none, one, or more instances in a process or a vnode.

Instances have simple data types: strings or strings that can be interpreted as integers.

An IMA table is modeled by having objects with instance values that form a key index.

MIB Creation

The IMA's SQL MIB is the binding of the MIB object space into the SQL name space. Tables containing MIB objects can be registered into any database owned by the user "$ingres". The binding is done by IMA, which converts between rows and individual MIB objects.

Registration of IMA Tables

To define an SQL schema for a MIB database, IMA tables (and the IMA catalogs if they do not exist) must be created by the $ingres user in a database owned by $ingres.

The SQL statement REGISTER TABLE...AS IMPORT is used to register IMA tables.

REGISTER TABLE Statement—Register IMA Table

The REGISTER TABLE...AS IMPORT statement has the following syntax:

```
REGISTER TABLE tablename
(col_name format [IS 'ext_format'] {, col_name format [IS 'ext_format']})
AS IMPORT FROM 'source'
WITH DBMS = IMA
[, STRUCTURE = NONE | [UNIQUE] SORTEDKEY]
[, KEY = (col_name [ASC] {, col_name})]
[, ROWS = nnnnnnnn]
[, [NO]DUPLICATES]
[, [NO]UPDATE]
[, [NO]JOURNALING]
;
```

where:

tablename

(Required) Is the name of the table, which must follow the standard naming convention.

col_name

(Required) Is the name of the column, which must follow the standard naming convention.

format

(Required) Is the data type of the column.

IS 'ext_format'

Specifies the management object being referenced, by extended format string name. This can be:

Classid

A reserved name identifying metadata.

VNODE

Column is vnode places only char type.

SERVER

Column is server places only char type.

CLASSID

Column is the classid name char type.

INSTANCE

Column is the instance value char type.

VALUE

Column is the char value of the {place, classid, instance} object.

PERMISSIONS

Column is the integer management object permission mask for the classid.

AS IMPORT FROM '*source*'

(Required) Indicates that the table being registered is for a gateway product. The value of *source* must be enclosed in single quotation marks, and is either "objects" or "tables," as described in IMA Table Types (see page 255):

IMPORT FROM 'objects'

IS '*ext_format*' clause values can be any of the following:

'VNODE'

'SERVER'

'CLASSID'

'INSTANCE'

'VALUE'

'PERMISSIONS'

These values are case sensitive and must be uppercase.

IMPORT FROM 'tables'

IS '*ext_format*' clause values can be any of the following:

'VNODE' or 'SERVER'

classid

These values are case sensitive: 'VNODE' or 'SERVER' must be uppercase and the classids must be in lowercase.

WITH

(Required) Introduces additional information about the table being imported.

DBMS = IMA

(Required) Indicates that the table is registered with IMA. The only possible value is IMA.

STRUCTURE =

(Required) Specifies whether the table is keyed or not. For IMA tables, the only valid structure is either:

SORTKEYED

UNIQUE SORTKEYED

KEY =

(Required) Specifies the column name(s) to use as the key.

For tables with IMPORT FROM 'objects', the key must contain a maximum of three columns from the following: VNODE or SERVER, CLASSID, INSTANCE, and the order must follow the same sequence.

For tables with IMPORT FROM 'tables', the key must contain a maximum of two columns. If two columns are specified, the first must be either VNODE or SERVER, both VNODE and SERVER cannot be specified.

ROWS = *nnnnnnnn*

(Optional) Specifies the approximate number of rows in the table. If omitted, a default of 1000 is used, which is used by the query optimizer as the number of rows in the table.

[NO]DUPLICATES

(Optional) Indicates whether the table can contain duplicate rows. If this parameter is omitted, duplicates will be considered.

[NO]UPDATE

(Optional) Indicates whether the table can be updated.

For tables with IMPORT FROM 'objects' can be UPDATE or NOUPDATE.

For tables with IMPORT FROM 'tables' must be NOUPDATE.

If this parameter is omitted, the default is NOUPDATE (read-only).

[NO]JOURNALING

(Optional) Indicates whether journaling is allowed. For IMA tables, only NOJOURNALING is allowed.

IMA Table Types

There are two types of IMA tables:

- Flat tables
- Cross-tab tables

Flat Tables

Flat tables:

- Have one object instance per row
- Can be updated
- Have the clause: IMPORT FROM 'objects'

An example of a flat table registration is:

```
REGISTER TABLE ima_mib_objects (
      server        VARCHAR(64)  NOT NULL NOT DEFAULT IS 'SERVER',
      classid       VARCHAR (64) NOT NULL NOT DEFAULT IS 'CLASSID',
      instance      VARCHAR (64) NOT NULL NOT DEFAULT IS 'INSTANCE',
      value         VARCHAR(64)  NOT NULL NOT DEFAULT IS 'VALUE',
      perms         INTEGER2     NOT NULL NOT DEFAULT IS 'PERMISSIONS'
)
AS IMPORT FROM 'objects'
WITH UPDATE,
DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (server, classid, instance)
;
```

Sample output from ima_mib_objects table may look like this:

```
SELECT server, classid, instance, value
FROM   ima_mib_objects;
```

server	classid	instance	value
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_id	00000000001	00000000032
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_id	00000000002	00000000020
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_id	00000000003	00000000003
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_id	00000000004	00000000031
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_id	00000000005	00000000030
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_id	00000000006	00000000010
. . .			
Another 32175 rows (approx)			

The above SQL by default will give only the available IMA classid values from the DBMS server to which the SQL session is currently connected.

To see all the IMA classid values available in the VNODE (that is, all Ingres servers running in that domain), it is necessary to extend the domain (see page 261).

Cross-tab Tables

Cross-tab tables:

- Have the clause: IMPORT FROM 'tables'

- The columns in the table registration can be in any order

- A place column is optional. If present, it is identified by having an "IS ext_format" of value VNODE or SERVER. If omitted the place defaults to the current server.

- All other columns must have an "IS ext_format" of classid.

There are objects that describe IMA; these objects can be seen by registering a cross-tab table as follows:

```
REGISTER TABLE    ima_mo_meta
(
     server        VARCHAR(64) NOT NULL NOT DEFAULT IS 'SERVER',
     classid       VARCHAR(64) NOT NULL NOT DEFAULT IS 'exp.glf.mo.meta.classid',
     oid           VARCHAR(8)  NOT NULL NOT DEFAULT IS 'exp.glf.mo.meta.oid',
     perms         INTEGER2    NOT NULL NOT DEFAULT IS 'exp.glf.mo.meta.perms',
     size          INTEGER2    NOT NULL NOT DEFAULT IS 'exp.glf.mo.meta.size',
     xindex        VARCHAR(64) NOT NULL NOT DEFAULT IS 'exp.glf.mo.meta.index'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (server, classid);
```

Note: The table and column names do not have to be as shown; the table name ima_mo_meta is used for convenience only.

- Selecting from ima_mo_meta as created above will by default only give the objects for the DBMS to which the SQL session is currently connected.

- To see all the IMA objects available in the VNODE, it is necessary to extend the domain (see page 261).

Sample output from ima_mo_meta table may look like this:

```
SELECT server, classid, xindex
FROM   ima_mo_meta;
```

server	classid	xindex
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_id	exp.adf.adg.dt_ix
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_ix	exp.adf.adg.dt_ix
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_name	exp.adf.adg.dt_ix
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.dt_stat	exp.adf.adg.dt_ix
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.fi_dtarg1	exp.adf.adg.fi_ix
GRFR6::/@II\\INGRES\\db0	exp.adf.adg.fi_dtarg2	exp.adf.adg.fi_ix
. . .		
Another 1150 rows (approx.)		

Another example of a cross-tab table registration is:

UNIX:

```
REGISTER TABLE  ima_dbms_servers (
server             VARCHAR(64) NOT NULL NOT DEFAULT IS
                   'SERVER',
listen_address     VARCHAR(64) NOT NULL NOT DEFAULT IS
                   'exp.gcf.gca.client.listen_address',
max_connections    INTEGER4 NOT NULL NOT DEFAULT IS
                   'exp.scf.scd.server.max_connections',
num_connections    INTEGER4 NOT NULL NOT DEFAULT IS
                   'exp.scf.scd.server.current_connections',
server_pid         INTEGER4 NOT NULL NOT DEFAULT IS
                   'exp.clf.unix.cs.srv_block.pid'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = SORTKEYED,
KEY = (server);
```

Windows:

```
REGISTER TABLE  ima_dbms_servers (
server                  VARCHAR(64) NOT NULL NOT DEFAULT IS
                        'SERVER',
listen_address          VARCHAR(64) NOT NULL NOT DEFAULT IS
                        'exp.gcf.gca.client.listen_address',
max_connections         INTEGER4 NOT NULL NOT DEFAULT IS
                        'exp.scf.scd.server.max_connections',
num_connections         INTEGER4 NOT NULL NOT DEFAULT IS
                        'exp.scf.scd.server.current_connections',
server_pid              INTEGER4 NOT NULL NOT DEFAULT IS
                        'exp.clf.nt.cs.srv_block.pid'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = SORTKEYED,
KEY = (server);
```

Check to see that the classids belong either to the same "xindex" or to no index. If the classids come from a mix of xindexes, probably no rows will be returned.

```
SELECT
classid,
xindex
FROM
ima_mo_meta
WHERE
classid in
(
     'exp.gcf.gca.client.listen_address',
     'exp.scf.scd.server.max_connections',
     'exp.scf.scd.server.current_connections',
     'exp.clf.nt.cs.srv_block.pid'
)
```

classid	xindex
exp clf.nt.cs.srv_block.pdf	
exp.gcf.gca.client.listen_address	exp.gcf.gca.client
exp.scf.scd.server.current_connections	
exp.scf.scd.server.max_connections	

Check to see that the classids exist in the MIB server domain. If a table is registered with invalid classids, no error message is generated (at both registration and run time). If an invalid classid is used, no rows are returned from the registered table.

This can be demonstrated from the above SQL, if the REGISTER TABLE with classid 'exp.clf.nt.cs.srv_block.pid' is registered on an Ingres installation on UNIX or Linux and a SELECT statement run against the table, no rows are returned.

The same applies if the 'exp.clf.unix.cs.srv_block.pid' classid is registered against an Ingres installation on Windows.

Executing the following SQL:

```
SELECT *
FROM ima_dbms_servers
```

on an Ingres installation on Windows where the domain has not been extended, gives:

Server	Listen address	Max connections	Num connections	Server PID
GRFR6::/@II\\INGRES\\fe0	II\\INGRES\\fe0	500	2	4064

In the Ingres home directory under (depending on the version of Ingres) ingres/bin or ingres/vdba, there are scripts that Ingres uses to create the database objects (such as tables, views, and database procedures) within the IMADB database. These are:

- makimau.sql (for UNIX)
- makiman.sql (for Windows)
- makimav.sql (for VMS)

There are also two example applications in the SIG directory, in the directories

- IMA (An IMA-based example program)
- IMP (An IMA-based IPM application)

Both of these applications come with SQL scripts to create IMA-based tables.

REMOVE TABLE Statement—Remove Table Registrations

Use the SQL statement REMOVE TABLE to remove the definition of an IMA table. This statement removes all catalog entries for the registered table, including any related views, integrities, and permits.

This statement has the following syntax:

```
remove table tablename;
```

where:

tablename

Is the table name for a registered IMA table.

IMA DBMSINFO Constants

The DBMSINFO function returns a string containing information about the current session. Three DBMSINFO constants are for use in queries involving IMA tables.

This statement has the following syntax:

```
DBMSINFO('request_name')
```

where:

request_name refers to one of the following:

ima_vnode

Returns the IMA MIB server vnode

ima_server

Returns the IMA MIB server address

ima_session

Returns the IMA MIB server session ID

For more information about using DBMSINFO, see the *SQL Reference Guide.*

Management Domains

A Management Domain consists of the IMA objects visible to a Management Application. This domain is session-specific and by default is the MIB server to which the session is connected. This is the only server of the current session that is managed. It can be a local server or a remote server connected through Ingres Net.

There are three objects (classid) that can be used to extend or reduce the size of the managed domain:

- 'exp.gwf.gwm.session.control.add_vnode'

- 'exp.gwf.gwm.session.control.del_vnode'

- 'exp.gwf.gwm.session.control.reset_domain'

To add a vnode to the domain it is necessary to update the classid in ima_mib_objects (or the table that has been defined to be the 'objects' table).

Extending the Domain in the Local Instance

The following SQL update extends the domain from the MIB server to all servers in the local instance of Ingres. This makes visible, for example, other DBMS servers, logging and locking objects, RCP and ACP:

```
UPDATE  ima_mib_objects
SET     value      = DBMSINFO('IMA_VNODE')
WHERE   classid    = 'exp.gwf.gwm.session.control.add_vnode'
AND     instance   = '0'
AND     server     = DBMSINFO('IMA_SERVER');
```

It is not necessary to use DBMSINFO('IMA_SERVER') to define the name of the MIB server to which the session is connected; DBMSINFO('IMA_SERVER') could be replaced by a program variable or a hard coded value, but DBMSINFO('IMA_SERVER') is used for ease and to make the SQL generic across operating systems.

As the above SQL is extending the local MIB server domain, it is necessary to set the value of the "value" column to the local VNODE, which can be retrieved through the use of a program variable with the value in, a hard coded value, or through DBMSINFO('IMA_VNODE').

Note: Updating the "value" column will not actually change what is in the column; it is an instruction to the MIB server to expose the other servers in that VNODE.

Below is an example of the count of the number of classid and instances done before and after the local domain is extended:

	Before Extending the Domain	After Extending the Domain
ima_mib_objects	29076	60247
ima_mo_meta	1150	2675

The actual values will change depending on the number of servers running, the number of sessions, and so on.

Extending the Domain to a Remote Instance

To extend the domain to remote instance of Ingres, the instance must be reachable through Ingres Net. This can be done using netutil or VDBA. The login to the remote instance can either be done using an installation password or to a suitably privileged Ingres user, for example, "root".

The following SQL update extends the domain to a remote VNODE called "GRFR6":

```
UPDATE  ima_mib_objects
SET     value        = 'GRFR6'
WHERE   classid      = 'exp.gwf.gwm.session.control.add_vnode'
AND     instance     = '0'
AND     server       = DBMSINFO('IMA_SERVER');
```

The value 'GRFR' can be put into a program variable. Note that server is still the local MIB server as it is its domain that is being extended.

Removing a Remote Instance Domain

To remove the "GRFR6" VNODE from the MIB server domain, it is necessary to update the classid 'exp.gwf.gwm.session.control.del_vnode'. Doing so does not actually change the value in the ima_mib_objects table, but is an instruction to remove the objects from the visible domain.

The following SQL update removes the remote instance domain:

```
UPDATE  ima_mib_objects
SET     value        = 'GRFR6'
WHERE   classid      = 'exp.gwf.gwm.session.control.del_vnode'
AND     instance     = '0'
AND     server       = DBMSINFO('IMA_SERVER');
```

Restoring the Domain

To return the domain for the MIB server to the original default, it is necessary to update the classid 'exp.gwf.gwm.session.control.reset_domain'. Doing so does not actually change the value in the ima_mib_objects table, but is an instruction to remove the objects from the visible domain.

The following SQL update restores the domain:

```
UPDATE ima_mib_objects
SET    value       = DBMSINFO('IMA_VNODE')
WHERE  classid     = 'exp.gwf.gwm.session.control.reset_domain'
AND    instance    = '0'
AND    server      = DBMSINFO('IMA_SERVER');
```

To simplify this process further, the IMADB database contains database procedures:

- ima_set_server_domain

  This procedure takes no parameters. It updates the "exp.gwf.gwm.session.control.reset_domain' object.

- ima_set_vnode_domain

  This procedure takes a parameter of a VNODE. It updates the 'exp.gwf.gwm.session.control.add_vnode' object.

Control Objects

In the same way that there are control objects that can be updated to add or delete a VNODE from a MIB server domain, there are other control objects that can be updated using the SQL statement UPDATE.

The following is a sample of some of the control objects and their effects:

exp.scf.scd.server.control.close	Sets a DBMS server closed
exp.scf.scs.scb_remove_session	Removes a session from a DBMS server
exp.scf.scd.server.control.open	Sets a DBMS server open
exp.scf.scd.server.control.shut	Sets a DBMS server shut
exp.scf.scd.server.control.start_sampler	Start sampling on a DBMS server
exp.scf.scd.server.control.stop_sampler	Stop sampling on a DBMS server
exp.scf.scd.server.control.stop	Stops a DBMS server
exp.scf.scd.server.control.crash	Causes a DBMS server to CRASH
exp.gcf.gcn.local_vnode	Set/unset the name of the local_vnode
exp.gcf.gcn.remote_vnode	Set/Unset the name of remote_vnode

To simplify using some of these control objects, the IMADB database contains the following database procedures:

- ima_close_server
- ima_drop_all_sessions
- ima_drop_sessions
- ima_open_server
- ima_remove_session
- ima_shut_server
- ima_start_sampler
- ima_stop_sampler

Restrictions in IMA Use

Following is a summary of current restrictions within IMA:

- Support for data types is limited in table registrations.
    - Only INTEGER, VARCHAR, and CHAR data types can be specified as data types in the REGISTER TABLE statement.
    - All columns must be NOT NULL NOT DEFAULT.
- The IMA supports full SQL but with the following restrictions:
    - There is no locking or lock ownership of IMA underlying objects.
    - There are no transactions for updates for IMA objects.
    - Updates take immediate action and cannot be rolled back.
    - All reads are "dirty." This means that querying the same object may return different values because the data underlying the record could have changed.
    - Rule triggering is unreliable. Rules are not triggered when the value of an object changes in a way unknown through SQL. This occurs to IMA related data as the installation operates. Thus, while updates that are performed through SQL can trigger rules, rule integrity cannot be guaranteed for IMA data.
- The group of defined objects supplied with the IMADB database is subject to change in subsequent releases.
- A cluster installation is visible to IMA as the installations on each node within a domain.

REGISTER TABLE Examples

This section gives examples of tables that can be registered against the IMA MIB. While the tables in these examples could be registered within the IMADB, we suggest that you use a separate database.

These tables are then used in Query Examples on IMA Tables (see page 270).

Examples: Domain Tables

The ima_session_domain table below shows the domain of SERVERS and VNODES visible to the MIB Server domain:

```
REGISTER TABLE  ima_session_domain
(
domplace VARCHAR(64) NOT NULL NOT DEFAULT IS 'exp.gwf.gwm.session.dom.index'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (domplace);
```

The ima_places table below shows all the servers known to the DBMS, but only in the local MIB Server domain, because when there is no VNODE or SERVER specified, the default is SERVER:

```
REGISTER TABLE  ima_places
(
place VARCHAR(64) NOT NULL NOT DEFAULT IS 'exp.gwf.gwm.places.index',
type I4 NOT NULL NOT DEFAULT IS 'exp.gwf.gwm.places.type',
class VARCHAR(64) NOT NULL NOT DEFAULT IS 'exp.gwf.gwm.servers.class',
flags I4 NOT NULL NOT DEFAULT IS 'exp.gwf.gwm.servers.flags'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (place);
```

Example: Table of SCF Sessions

The ima_server_sessions table below shows information about SCF-level sessions:

```
REGISTER TABLE  ima_server_sessions
(
server VARCHAR(64) NOT NULL NOT DEFAULT IS 'SERVER',
session_id VARCHAR(32) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_index',
effective_user VARCHAR(20) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_euser',
real_user VARCHAR(20) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_ruser',
db_name VARCHAR(12) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_database',
db_owner VARCHAR(8) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_dbowner',
db_lock VARCHAR(9) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_dblockmode',
server_facility VARCHAR(10) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_facility_name',
session_activity VARCHAR(50) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_activity',
activity_detail VARCHAR(50) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_act_detail',
session_query VARCHAR(1000) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_query',
session_terminal VARCHAR(12) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_terminal',
session_group VARCHAR(8) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_group',
session_role VARCHAR(8) NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_role',
server_pid INTEGER4 NOT NULL NOT DEFAULT IS 'exp.scf.scs.scb_pid'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (server, session_id)
```

Example: Lock Tables

The ima_locklists, ima_locks, and ima_resources tables below show information about locks and the resources those locks are for.

Note: Because the underlying locks are rapidly changing, data in these IMA tables are susceptible to "dirty reads." This means that querying the same object may return different values due to the underlying data having changed.

```
REGISTER TABLE  ima_locklists
(
vnode VARCHAR(64) NOT NULL NOT DEFAULT IS 'VNODE',
locklist_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_id.id_id',
locklist_lkb_count INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_lkb_count',
locklist_status VARCHAR(50) NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_status',
locklist_lock_count INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_llkb_count',
locklist_max_locks INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_max_lkb',
locklist_wait_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_wait_id_id',
locklist_type INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_name0',
locklist_database INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_name1',
locklist_server_pid INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_pid',
locklist_session_id VARCHAR(32) NOT NULL NOT DEFAULT IS 'exp.dmf.lk.llb_sid'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (vnode, locklist_id_id);

REGISTER TABLE  ima_locks
(
vnode VARCHAR(64) NOT NULL NOT DEFAULT IS 'VNODE',
lock_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.lkb_id.id_id',
lock_request_mode VARCHAR(10) NOT NULL NOT DEFAULT IS 'exp.dmf.lk.lkb_request_mode',
lock_grant_mode VARCHAR(10) NOT NULL NOT DEFAULT IS 'exp.dmf.lk.lkb_grant_mode',
lock_state VARCHAR(100) NOT NULL NOT DEFAULT IS 'exp.dmf.lk.lkb_state',
lock_attributes VARCHAR(100) NOT NULL NOT DEFAULT IS 'exp.dmf.lk.lkb_attribute',
resource_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.lkb_rsb_id_id',
locklist_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.lkb_llb_id_id'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (vnode, lock_id_id);
```

Note: In the ima_locks table, state 1 = granted, 2 = convert and 3 = waiting.

```
REGISTER TABLE  ima_resources
(
vnode VARCHAR(64) NOT NULL NOT DEFAULT IS 'VNODE',
resource_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.rsb_id.id_id',
resource_grant_mode VARCHAR(4) NOT NULL NOT DEFAULT IS 'exp.dmf.lk.rsb_grant_mode',
resource_key VARCHAR(50) NOT NULL NOT DEFAULT IS 'exp.dmf.lk.rsb_name',
resource_type INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.rsb_name0',
resource_database_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.rsb_name1',
```

```
resource_table_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.rsb_name2',
resource_index_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.rsb_name3',
resource_page_number INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lk.rsb_name4'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (vnode, resource_id_id);
```

Example: Logging Tables

The ima_log_processes, ima_log_databases and ima_log_transactions tables below show information about transactions on the system. The data is susceptible to dirty reads:

```
REGISTER TABLE  ima_log_processes
(
vnode VARCHAR(64) NOT NULL NOT DEFAULT IS 'VNODE',
process_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lpb_id.id_id',
process_id_instance INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lpb_id.id_instance',
process_status VARCHAR(100) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lpb_status',
process_pid INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lpb_pid'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (vnode, process_id_id);

REGISTER TABLE  ima_log_databases
(
vnode VARCHAR(64) is 'VNODE',
db_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.ldb_id.id_id',
db_id_instance INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.ldb_id.id_instance',
db_status VARCHAR(100) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.ldb_status',
db_database_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.ldb_database_id',
db_name VARCHAR(32) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.ldb_db_name',
db_owner VARCHAR(32) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.ldb_db_owner'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (vnode, db_id_id);
```

```
REGISTER TABLE    ima_log_transactions
(
vnode VARCHAR(64) NOT NULL NOT DEFAULT IS 'VNODE',
tx_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_id.id_id',
tx_id_instance INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_id.id_instance',
tx_status VARCHAR(100) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_status',
tx_db_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_db_id_id',
tx_db_name VARCHAR(32) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_db_name',
tx_db_owner VARCHAR(32) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_db_owner',
tx_pr_id_id INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_pr_id_id',
tx_wait_reason VARCHAR(30) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_wait_reason',
tx_first_log_address VARCHAR (30) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_first_lga',
tx_last_log_address VARCHAR (30) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_last_lga',
tx_transaction_id VARCHAR(16) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_tran_id',
tx_transaction_high INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_tran_id.db_high_tran',
tx_transaction_low INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_tran_id.db_low_tran',
tx_server_pid INTEGER4 NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_pid',
tx_session_id VARCHAR(32) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_sid',
tx_user_name VARCHAR(32) NOT NULL NOT DEFAULT IS 'exp.dmf.lg.lxb_user_name'
)
AS IMPORT FROM 'tables'
WITH DBMS = IMA,
STRUCTURE = UNIQUE SORTKEYED,
KEY = (vnode, tx_id_id);
```

Query Examples on IMA Tables

This section provides example queries based on the tables registered in the REGISTER TABLE Examples (see page 266).

Note: The VNODE "GRFR6" is an example name for a VNODE.

Note: It is assumed that each example has been run in a new session.

Example: What Are User Sessions Doing?

The following SQL shows what user sessions are currently doing in all servers in the local VNODE and the remote VNODE "GRFR6":

```
update ima_mib_objects
set     value       = DBMSINFO('IMA_VNODE')
where   classid     = 'exp.gwf.gwm.session.control.add_vnode'
and     instance    = '0'
and     server      = DBMSINFO('IMA_SERVER');

update ima_mib_objects
set     value       = 'GRFR6'
where   classid     = 'exp.gwf.gwm.session.control.add_vnode'
and     instance    = '0'
and     server      = DBMSINFO('IMA_SERVER');

select
        *
from
        ima_server_sessions
where db_owner != '';
```

Example: Who Is Waiting for a Lock?

The following query shows which sessions are waiting for locks in the VNODE "GRFR6":

```
update ima_mib_objects
set     value       = 'GRFR6'
where   classid     = 'exp.gwf.gwm.session.control.add_vnode'
and     instance    = '0'
and     server      = DBMSINFO('IMA_SERVER');

select distinct
        resource_id_id,
        lock_id_id,
        lock_state,
        ima_locklists.locklist_id_id,
        locklist_server_pid,
        locklist_session_id,
        effective_user,
        db_name,
        session_terminal,
        session_query
from
        ima_locks,
        ima_locklists,
        ima_server_sessions
where lock_state != 'GR'
and ima_locks.locklist_id_id = ima_locklists.locklist_id_id
and ima_locklists.locklist_server_pid = ima_server_sessions.server_pid
and ima_locklists.locklist_session_id = ima_server_sessions.session_id;
```

Example: Who Is Holding Locks?

The following query shows which sessions are holding locks on the local VNODE:

```
update ima_mib_objects
set     value       = DBMSINFO('IMA_VNODE')
where   classid     = 'exp.gwf.gwm.session.control.add_vnode'
and     instance    = '0'
and     server      = DBMSINFO('IMA_SERVER');

select distinct
        lock_id_id,
        lock_state,
        ima_locklists.locklist_id_id,
        locklist_server_pid,
        locklist_session_id,
        effective_user,
        db_name,
        session_terminal,
        session_query
from
        ima_locks,
        ima_locklists,
        ima_resources,
        ima_server_sessions
where lock_state != 'WT'
and ima_locks.locklist_id_id = ima_locklists.locklist_id_id
and ima_locklists.locklist_server_pid = ima_server_sessions.server_pid
and ima_locklists.locklist_session_id = ima_server_sessions.session_id;
```

Example: Who Is Holding a Lock that Other Sessions Need?

The following query shows which sessions on the local VNODE are holding locks where other sessions are waiting:

```
update  ima_mib_objects
set     value        = DBMSINFO('IMA_VNODE')
where   classid      = 'exp.gwf.gwm.session.control.add_vnode'
and     instance     = '0'
and     server       = DBMSINFO('IMA_SERVER');

select distinct
        resource_id_id,
        lock_id_id,
        lock_state,
        ima_locklists.locklist_id_id,
        locklist_server_pid,
        locklist_session_id,
        effective_user,
        db_name,
        session_terminal
        from
        ima_locks,
        ima_locklists,
        ima_server_sessions
where resource_id_id in
        (
        select distinct
                ima_locks.resource_id_id
        from
                ima_locks
        where lock_state != 'GR'
        )
and lock_state != 'WT'
and ima_locks.locklist_id_id = ima_locklists.locklist_id_id
and ima_locklists.locklist_server_pid = ima_server_sessions.server_pid
and ima_locklists.locklist_session_id = ima_server_sessions.session_id;
```

Example: Shutting Down a DBMS Server

This query marks the MIB server for shutdown when all sessions have exited:

Windows:

```
update ima_mib_objects
set value    = '1'
where classid = 'exp.clf.nt.cs.mon.shutserver'
and instance  = '0'
and server    = DBMSINFO('IMA_SERVER');
```

UNIX:

```
update ima_mib_objects
set value    = '1'
where classid = 'exp.clf.unix.cs.mon.shutserver'
and instance  = '0'
and server    = DBMSINFO('IMA_SERVER');
```

Class ID Objects

This list of classid objects is taken from an Ingres 2006 9.0.4 release.

On Windows XP, the following servers were running:

- DBMS
- GCC
- Star
- RCP
- ACP
- Bridge Server
- DAS
- JDBC
- GCN
- Remote Command server

On an Ingres 2006 9.0.4 NPTL release on Linux, the following servers were running:

- DBMS
- GCC
- Star
- RCP
- ACP
- DAS
- GCN

Notes:

1. This is not a complete list of all IMA objects in Ingres.

2. A description is provided, where time has allowed.

3. Objects with IMA can be added or removed.

Classid	Description
exp.adf.adg.dt_id	
exp.adf.adg.dt_ix	
exp.adf.adg.dt_name	
exp.adf.adg.dt_stat	

exp.adf.adg.fi_dtarg1	
exp.adf.adg.fi_dtarg2	
exp.adf.adg.fi_dtarg3	
exp.adf.adg.fi_dtarg4	
exp.adf.adg.fi_dtresult	
exp.adf.adg.fi_id	
exp.adf.adg.fi_ix	
exp.adf.adg.fi_numargs	
exp.adf.adg.fi_opid	
exp.adf.adg.fi_type	
exp.adf.adg.op_id	
exp.adf.adg.op_ix	
exp.adf.adg.op_name	
exp.adf.adg.op_type	
exp.clf.di.di_slaveno	The UNIX server slave number, starting at 1. The processes themselves are numbered from 0.
exp.clf.di.dimo_collect	Setting this object causes DI I/O slave statistics to be collected.
exp.clf.di.dimo_cpu.tm_msecs	Number of milliseconds of CPU consumed by the slave
exp.clf.di.dimo_cpu.tm_secs	Number of seconds of CPU consumed by the slave
exp.clf.di.dimo_dio	Number of DIO (disk I/O) operations done by the slave
exp.clf.di.dimo_idrss	The resident set size of the slave
exp.clf.di.dimo_majflt	Number of major page faults in the slave
exp.clf.di.dimo_maxrss	The maximum resident set size of the slave
exp.clf.di.dimo_minflt	Number of minor page faults by the slave
exp.clf.di.dimo_msgrcv	Number of messages received by the slave
exp.clf.di.dimo_msgsnd	Number of messages sent by the slave
exp.clf.di.dimo_msgtotal	Total number of messages sent and received by the slave
exp.clf.di.dimo_nivcsw	Number of involuntary context switches by the slave

exp.clf.di.dimo_nsignals	Number of signals received by the slave
exp.clf.di.dimo_nswap	Number of times the slave has been swapped out
exp.clf.di.dimo_nvcsw	Number of voluntary context switches out of the slave (probably for i/o)
exp.clf.di.dimo_reads	Number of read operations done by the slave
exp.clf.di.dimo_stime.tm_msecs	System CPU milliseconds in the slave
exp.clf.di.dimo_stime.tm_secs	System CPU seconds consumed by the slave
exp.clf.di.dimo_utime.tm_msecs	User CPU milliseconds by the slave
exp.clf.di.dimo_utime.tm_secs	User CPU seconds by the slave
exp.clf.di.dimo_writes	Number of write calls by the slave
exp.clf.gv.bldlevel	
exp.clf.gv.bytetype	
exp.clf.gv.charset	
exp.clf.gv.cnf_index	
exp.clf.gv.cnf_name	
exp.clf.gv.cnf_value	
exp.clf.gv.cnfdat_index	
exp.clf.gv.cnfdat_name	
exp.clf.gv.cnfdat_value	
exp.clf.gv.env	
exp.clf.gv.genlevel	
exp.clf.gv.hw	
exp.clf.gv.instance	
exp.clf.gv.language	
exp.clf.gv.majorvers	
exp.clf.gv.minorvers	
exp.clf.gv.os	
exp.clf.gv.patchlvl	
exp.clf.gv.sql92_conf	
exp.clf.gv.system	
exp.clf.gv.tcpport	

exp.clf.gv.version
exp.clf.nt.cs.cnd_index
exp.clf.nt.cs.cnd_name
exp.clf.nt.cs.cnd_next
exp.clf.nt.cs.cnd_prev
exp.clf.nt.cs.cnd_waiter
exp.clf.nt.cs.desched_usec_sleep
exp.clf.nt.cs.dio_resumes
exp.clf.nt.cs.lastquant
exp.clf.nt.cs.max_sem_loops
exp.clf.nt.cs.mon.breakpoint
exp.clf.nt.cs.mon.crashserver
exp.clf.nt.cs.mon.debug
exp.clf.nt.cs.mon.resume_session
exp.clf.nt.cs.mon.rm_session
exp.clf.nt.cs.mon.shutserver
exp.clf.nt.cs.mon.stopcond
exp.clf.nt.cs.mon.stopserver
exp.clf.nt.cs.mon.suspend_session
exp.clf.nt.cs.num_processors
exp.clf.nt.cs.samp_index
exp.clf.nt.cs.samp_numsamples
exp.clf.nt.cs.samp_numtlsdirty
exp.clf.nt.cs.samp_numtlsprobes
exp.clf.nt.cs.samp_numtlsreads
exp.clf.nt.cs.samp_numtlssamples
exp.clf.nt.cs.samp_numtlsslots
exp.clf.nt.cs.samp_numtlswrites
exp.clf.nt.cs.scb_bio
exp.clf.nt.cs.scb_client_type
exp.clf.nt.cs.scb_cnd
exp.clf.nt.cs.scb_connect

exp.clf.nt.cs.scb_cputime
exp.clf.nt.cs.scb_dio
exp.clf.nt.cs.scb_index
exp.clf.nt.cs.scb_inkernel
exp.clf.nt.cs.scb_length
exp.clf.nt.cs.scb_lio
exp.clf.nt.cs.scb_locks
exp.clf.nt.cs.scb_mask
exp.clf.nt.cs.scb_mask_num
exp.clf.nt.cs.scb_memory
exp.clf.nt.cs.scb_mode
exp.clf.nt.cs.scb_nmode
exp.clf.nt.cs.scb_owner
exp.clf.nt.cs.scb_pid
exp.clf.nt.cs.scb_ppid
exp.clf.nt.cs.scb_self
exp.clf.nt.cs.scb_state
exp.clf.nt.cs.scb_state_num
exp.clf.nt.cs.scb_stk_size
exp.clf.nt.cs.scb_svcb
exp.clf.nt.cs.scb_thread_id
exp.clf.nt.cs.scb_thread_type
exp.clf.nt.cs.scb_thread_type_num
exp.clf.nt.cs.scb_type
exp.clf.nt.cs.scb_username
exp.clf.nt.cs.srv_block.aq_length
exp.clf.nt.cs.srv_block.aquantum
exp.clf.nt.cs.srv_block.attn
exp.clf.nt.cs.srv_block.bquantum0
exp.clf.nt.cs.srv_block.bquantum1
exp.clf.nt.cs.srv_block.cpu
exp.clf.nt.cs.srv_block.current

exp.clf.nt.cs.srv_block.cursors
exp.clf.nt.cs.srv_block.disconnect
exp.clf.nt.cs.srv_block.elog
exp.clf.nt.cs.srv_block.error_code
exp.clf.nt.cs.srv_block.event_mask
exp.clf.nt.cs.srv_block.format
exp.clf.nt.cs.srv_block.gca_name
exp.clf.nt.cs.srv_block.hwm_active
exp.clf.nt.cs.srv_block.idle_time
exp.clf.nt.cs.srv_block.known_list
exp.clf.nt.cs.srv_block.mask
exp.clf.nt.cs.srv_block.max_active
exp.clf.nt.cs.srv_block.max_sessions
exp.clf.nt.cs.srv_block.next_id
exp.clf.nt.cs.srv_block.num_active
exp.clf.nt.cs.srv_block.num_sessions
exp.clf.nt.cs.srv_block.pid
exp.clf.nt.cs.srv_block.process
exp.clf.nt.cs.srv_block.q_per_sec
exp.clf.nt.cs.srv_block.quantums
exp.clf.nt.cs.srv_block.read
exp.clf.nt.cs.srv_block.ready_mask
exp.clf.nt.cs.srv_block.reject
exp.clf.nt.cs.srv_block.saddr
exp.clf.nt.cs.srv_block.scballoc
exp.clf.nt.cs.srv_block.scbdealloc
exp.clf.nt.cs.srv_block.sem_stats.cs_smms_count
exp.clf.nt.cs.srv_block.sem_stats.smmsx_count
exp.clf.nt.cs.srv_block.sem_stats.smmx_count
exp.clf.nt.cs.srv_block.sem_stats.smmxx_count
exp.clf.nt.cs.srv_block.sem_stats.smss_count
exp.clf.nt.cs.srv_block.sem_stats.smssx_count

exp.clf.nt.cs.srv_block.sem_stats.smsx_count
exp.clf.nt.cs.srv_block.sem_stats.smsxx_count
exp.clf.nt.cs.srv_block.shutdown
exp.clf.nt.cs.srv_block.startup
exp.clf.nt.cs.srv_block.state
exp.clf.nt.cs.srv_block.stk_count
exp.clf.nt.cs.srv_block.stk_list
exp.clf.nt.cs.srv_block.stksize
exp.clf.nt.cs.srv_block.svcb
exp.clf.nt.cs.srv_block.to_list
exp.clf.nt.cs.srv_block.toq_cnt
exp.clf.nt.cs.srv_block.user_sessions
exp.clf.nt.cs.srv_block.wait_stats.bio_done
exp.clf.nt.cs.srv_block.wait_stats.bio_idle
exp.clf.nt.cs.srv_block.wait_stats.bio_time
exp.clf.nt.cs.srv_block.wait_stats.bio_waits
exp.clf.nt.cs.srv_block.wait_stats.bior_done
exp.clf.nt.cs.srv_block.wait_stats.bior_time
exp.clf.nt.cs.srv_block.wait_stats.bior_waits
exp.clf.nt.cs.srv_block.wait_stats.biow_done
exp.clf.nt.cs.srv_block.wait_stats.biow_time
exp.clf.nt.cs.srv_block.wait_stats.biow_waits
exp.clf.nt.cs.srv_block.wait_stats.dio_done
exp.clf.nt.cs.srv_block.wait_stats.dio_idle
exp.clf.nt.cs.srv_block.wait_stats.dio_time
exp.clf.nt.cs.srv_block.wait_stats.dio_waits
exp.clf.nt.cs.srv_block.wait_stats.dior_done
exp.clf.nt.cs.srv_block.wait_stats.dior_time
exp.clf.nt.cs.srv_block.wait_stats.dior_waits
exp.clf.nt.cs.srv_block.wait_stats.diow_done
exp.clf.nt.cs.srv_block.wait_stats.diow_time
exp.clf.nt.cs.srv_block.wait_stats.diow_waits

exp.clf.nt.cs.srv_block.wait_stats.lg_done
exp.clf.nt.cs.srv_block.wait_stats.lg_idle
exp.clf.nt.cs.srv_block.wait_stats.lg_time
exp.clf.nt.cs.srv_block.wait_stats.lg_waits
exp.clf.nt.cs.srv_block.wait_stats.lge_done
exp.clf.nt.cs.srv_block.wait_stats.lge_time
exp.clf.nt.cs.srv_block.wait_stats.lge_waits
exp.clf.nt.cs.srv_block.wait_stats.lio_done
exp.clf.nt.cs.srv_block.wait_stats.lio_time
exp.clf.nt.cs.srv_block.wait_stats.lio_waits
exp.clf.nt.cs.srv_block.wait_stats.lior_done
exp.clf.nt.cs.srv_block.wait_stats.lior_time
exp.clf.nt.cs.srv_block.wait_stats.lior_waits
exp.clf.nt.cs.srv_block.wait_stats.liow_done
exp.clf.nt.cs.srv_block.wait_stats.liow_time
exp.clf.nt.cs.srv_block.wait_stats.liow_waits
exp.clf.nt.cs.srv_block.wait_stats.lk_done
exp.clf.nt.cs.srv_block.wait_stats.lk_idle
exp.clf.nt.cs.srv_block.wait_stats.lk_time
exp.clf.nt.cs.srv_block.wait_stats.lk_waits
exp.clf.nt.cs.srv_block.wait_stats.lke_done
exp.clf.nt.cs.srv_block.wait_stats.lke_time
exp.clf.nt.cs.srv_block.wait_stats.lke_waits
exp.clf.nt.cs.srv_block.wait_stats.tm_done
exp.clf.nt.cs.srv_block.wait_stats.tm_idle
exp.clf.nt.cs.srv_block.wait_stats.tm_time
exp.clf.nt.cs.srv_block.wait_stats.tm_waits
exp.clf.nt.cs.srv_block.write
exp.clf.nt.cs.srv_block.wt_list
exp.clf.nt.cs.thread_eventDISKIO
exp.clf.nt.cs.thread_eventLOCK
exp.clf.nt.cs.thread_eventLOCKEVENT

exp.clf.nt.cs.thread_eventLOG
exp.clf.nt.cs.thread_eventLOGEVENT
exp.clf.nt.cs.thread_eventLOGIO
exp.clf.nt.cs.thread_eventMESSAGEIO
exp.clf.nt.cs.thread_facilityADF
exp.clf.nt.cs.thread_facilityCLF
exp.clf.nt.cs.thread_facilityDMF
exp.clf.nt.cs.thread_facilityDUF
exp.clf.nt.cs.thread_facilityGCF
exp.clf.nt.cs.thread_facilityGWF
exp.clf.nt.cs.thread_facilityOPF
exp.clf.nt.cs.thread_facilityPSF
exp.clf.nt.cs.thread_facilityQEF
exp.clf.nt.cs.thread_facilityQSF
exp.clf.nt.cs.thread_facilityRDF
exp.clf.nt.cs.thread_facilityRQF
exp.clf.nt.cs.thread_facilitySCF
exp.clf.nt.cs.thread_facilitySXF
exp.clf.nt.cs.thread_facilityTPF
exp.clf.nt.cs.thread_facilityULF
exp.clf.nt.cs.thread_numsamples
exp.clf.nt.cs.thread_stateCNDWAIT
exp.clf.nt.cs.thread_stateCOMPUTABLE
exp.clf.nt.cs.thread_stateEVENT_WAIT
exp.clf.nt.cs.thread_stateFREE
exp.clf.nt.cs.thread_stateMUTEX
exp.clf.nt.cs.thread_stateSTACK_WAIT
exp.clf.nt.cs.thread_stateUWAIT
exp.clf.nt.me.num.bytes_used
exp.clf.nt.me.num.free_pages
exp.clf.nt.me.num.get_pages
exp.clf.nt.me.num.pages_used

exp.clf.nt.me.num.test_count	
exp.clf.nt.me.num.test_errs	
exp.clf.nt.me.num.test_mod	
exp.clf.unix.cs.cnd_index	ID of a condition variable, a string of the decimal address of the variable. Since this may be a 64-bit pointer, it cannot be held in an integer
exp.clf.unix.cs.cnd_name	The user-given name for the condition variable
exp.clf.unix.cs.cnd_next	The next condition in the chain
exp.clf.unix.cs.cnd_prev	The previous condition in the chain
exp.clf.unix.cs.cnd_waiter	The SCB waiting for the condition
exp.clf.unix.cs.desched_usec_sleep	On uni-processor, amount of time to sleep to de-schedule this server (in microseconds), allowing the server holding a CS semaphore to run (and presumably release it); controlled by the variable II_DESCHED_USEC_SLEEP
exp.clf.unix.cs.dio_resumes	
exp.clf.unix.cs.lastquant	
exp.clf.unix.cs.max_sem_loops	Maximum spin loops for CS semaphore before sleeping begins; controlled by the variable II_MAX_SEM_LOOPS
exp.clf.unix.cs.mon.breakpoint	
exp.clf.unix.cs.mon.crashserver	
exp.clf.unix.cs.mon.debug	
exp.clf.unix.cs.mon.resume_session	
exp.clf.unix.cs.mon.rm_session	OBSOLETE. This object is dangerous. Use exp.scf.scs.scb_remove_session.
exp.clf.unix.cs.mon.shutserver	
exp.clf.unix.cs.mon.stopcond	
exp.clf.unix.cs.mon.stopserver	
exp.clf.unix.cs.mon.suspend_session	
exp.clf.unix.cs.num_processors	Number of processors believed to exist on this host. Used to pick algorithms appropriate for different environments. Use II_NUM_OF_PROCESSORS to change.

exp.clf.unix.cs.scb_asmask	Thread's async mask as a string
exp.clf.unix.cs.scb_asmask_num	Async mask–look up ima_async_mask in IMA_STATUS_LOOKUP
exp.clf.unix.cs.scb_asunmask	
exp.clf.unix.cs.scb_asunmask_num	
exp.clf.unix.cs.scb_async	
exp.clf.unix.cs.scb_base_priority	
exp.clf.unix.cs.scb_bio	Number of "buffered" I/Os charged to this session
exp.clf.unix.cs.scb_bior	
exp.clf.unix.cs.scb_biow	
exp.clf.unix.cs.scb_client_type	Zero in SCB's for CL-level threads. The client's identification field.
exp.clf.unix.cs.scb_cnd	Condition being waited for, if any
exp.clf.unix.cs.scb_connect	Time in seconds from epoch when session was started.
exp.clf.unix.cs.scb_cputime	Seconds of CPU charged to this thread. May be inaccurate.
exp.clf.unix.cs.scb_dio	Disk I/O charged to this thread
exp.clf.unix.cs.scb_dior	
exp.clf.unix.cs.scb_diow	
exp.clf.unix.cs.scb_ef_mask	Event flag wait mask
exp.clf.unix.cs.scb_index	The ID of the session, a decimal string of the address of the control block. Since this might be a 64-bit pointer, it cannot fit into an integer.
exp.clf.unix.cs.scb_inkernel	Flag indicating whether in delicate internal state (switching thread, checking quantum, and so on)
exp.clf.unix.cs.scb_length	Actual length of this SCB, as allocated.
exp.clf.unix.cs.scb_lgevent	
exp.clf.unix.cs.scb_lio	
exp.clf.unix.cs.scb_lior	
exp.clf.unix.cs.scb_liow	
exp.clf.unix.cs.scb_lkevent	

exp.clf.unix.cs.scb_locks	OBSOLETE
exp.clf.unix.cs.scb_logs	
exp.clf.unix.cs.scb_mask	String of session mask
exp.clf.unix.cs.scb_mask_num	Bitmap of session mask–look up ima_async_mask in IMA_STATUS_LOOKUP
exp.clf.unix.cs.scb_memory	OBSOLETE
exp.clf.unix.cs.scb_mode	Session mode–look up ima_session_modes in IMA_STATUS_LOOKUP
exp.clf.unix.cs.scb_nmode	Next session mode–look up ima_session_modes in IMA_STATUS_LOOKUP
exp.clf.unix.cs.scb_owner	Constant 0xAB0000BA
exp.clf.unix.cs.scb_pid	PID of the server containing this session
exp.clf.unix.cs.scb_ppid	
exp.clf.unix.cs.scb_priority	Current thread priority
exp.clf.unix.cs.scb_self	CS_SID of the SCB. The same as the index in the UNIX CL.
exp.clf.unix.cs.scb_sem_count	Number of semaphores locked by this thread
exp.clf.unix.cs.scb_sm_next	First semaphore owned by this thread
exp.clf.unix.cs.scb_state	One of CS_FREE, CS_COMPUTABLE, CS_STACK_WAIT, CS_UWAIT, CS_EVENT_WAIT, CS_MUTEX, CS_CNDWAIT
exp.clf.unix.cs.scb_state_num	Thread state–look up ima_thread_state in IMA_STATUS_LOOKUP
exp.clf.unix.cs.scb_stk_area	OBSOLETE
exp.clf.unix.cs.scb_stk_size	Stack size
exp.clf.unix.cs.scb_svcb	Pointer to server block that started this thread
exp.clf.unix.cs.scb_thread_id	
exp.clf.unix.cs.scb_thread_type	-1 for internal thread, 0 for user thread
exp.clf.unix.cs.scb_thread_type_num	-1 for internal thread, 0 for user thread
exp.clf.unix.cs.scb_timeout	Current timeout value, in seconds
exp.clf.unix.cs.scb_type	Constant 0xABCD
exp.clf.unix.cs.scb_uic	The UID of the session owner; mostly useless because it is not really known and

	meaningless default values are used
exp.clf.unix.cs.scb_username	The username associated with this thread, from the connection information
exp.clf.unix.cs.srv_block.aq_length	
exp.clf.unix.cs.srv_block.aquantum	
exp.clf.unix.cs.srv_block.attn	
exp.clf.unix.cs.srv_block.bquantum0	
exp.clf.unix.cs.srv_block.bquantum1	
exp.clf.unix.cs.srv_block.cpu	CPU time (seconds) in process so far
exp.clf.unix.cs.srv_block.current	Currently active session ID
exp.clf.unix.cs.srv_block.cursors	Configured number of active cursors per session
exp.clf.unix.cs.srv_block.disconnect	
exp.clf.unix.cs.srv_block.elog	
exp.clf.unix.cs.srv_block.error_code	Current "major" error STATUS in the server
exp.clf.unix.cs.srv_block.event_mask	
exp.clf.unix.cs.srv_block.facility	
exp.clf.unix.cs.srv_block.format	
exp.clf.unix.cs.srv_block.gca_name	The GCA listen address for this server
exp.clf.unix.cs.srv_block.hwm_active	
exp.clf.unix.cs.srv_block.idle_time	
exp.clf.unix.cs.srv_block.known_list	
exp.clf.unix.cs.srv_block.mask	
exp.clf.unix.cs.srv_block.max_active	Max number of active threads allowed
exp.clf.unix.cs.srv_block.max_sessions	Max number of threads allowed
exp.clf.unix.cs.srv_block.next_id	
exp.clf.unix.cs.srv_block.num_active	Number of currently active threads
exp.clf.unix.cs.srv_block.num_sessions	Number of current sessions
exp.clf.unix.cs.srv_block.pid	OBSOLETE. Use exp.scf.scd.server.pid
exp.clf.unix.cs.srv_block.process	
exp.clf.unix.cs.srv_block.q_per_sec	Number of quantums per second
exp.clf.unix.cs.srv_block.quantums	Number of elapsed quantum intervals
exp.clf.unix.cs.srv_block.read	

exp.clf.unix.cs.srv_block.ready_mask	
exp.clf.unix.cs.srv_block.reject	
exp.clf.unix.cs.srv_block.saddr	
exp.clf.unix.cs.srv_block.scballoc	
exp.clf.unix.cs.srv_block.scbdealloc	
exp.clf.unix.cs.srv_block.sem_stats.cs_smmp_count	Number of multi-processor naps
exp.clf.unix.cs.srv_block.sem_stats.cs_smms_count	
exp.clf.unix.cs.srv_block.sem_stats.cs_smnonserver_count	Number of non-server naps
exp.clf.unix.cs.srv_block.sem_stats.cs_smsp_count	Number of single processor naps
exp.clf.unix.cs.srv_block.sem_stats.smc_count	Requests for cross-process semaphores
exp.clf.unix.cs.srv_block.sem_stats.smcl_count	Spins waiting for cross-process semaphores
exp.clf.unix.cs.srv_block.sem_stats.smcx_count	Collisions on cross-process semaphores
exp.clf.unix.cs.srv_block.sem_stats.smmsx_count	
exp.clf.unix.cs.srv_block.sem_stats.smmx_count	
exp.clf.unix.cs.srv_block.sem_stats.smmxx_count	
exp.clf.unix.cs.srv_block.sem_stats.sms_count	Requests for shared semaphore
exp.clf.unix.cs.srv_block.sem_stats.smss_count	
exp.clf.unix.cs.srv_block.sem_stats.smssx_count	
exp.clf.unix.cs.srv_block.sem_stats.smsx_count	Collisions on shared versus exclusive
exp.clf.unix.cs.srv_block.sem_stats.smsxx_count	
exp.clf.unix.cs.srv_block.sem_stats.smx_count	Requests for exclusive semaphore access
exp.clf.unix.cs.srv_block.sem_stats.smxx_count	Collisions exclusive versus exclusive
exp.clf.unix.cs.srv_block.shutdown	
exp.clf.unix.cs.srv_block.startup	
exp.clf.unix.cs.srv_block.state	Server internal state–look up using ima_server_internal_state_mask in IMA_STATUS_LOOKUP
exp.clf.unix.cs.srv_block.stk_count	
exp.clf.unix.cs.srv_block.stk_list	
exp.clf.unix.cs.srv_block.stksize	Configured stack size
exp.clf.unix.cs.srv_block.svcb	
exp.clf.unix.cs.srv_block.to_list	List of SCB's waiting for events that can

	time out
exp.clf.unix.cs.srv_block.toq_cnt	
exp.clf.unix.cs.srv_block.user_sessions	How many non-CS threads there are
exp.clf.unix.cs.srv_block.wait_stats.bio_done	Total BIOs done
exp.clf.unix.cs.srv_block.wait_stats.bio_idle	BIOs while idle
exp.clf.unix.cs.srv_block.wait_stats.bio_time	Tick count in BIO wait
exp.clf.unix.cs.srv_block.wait_stats.bio_waits	Number of waits for BIO
exp.clf.unix.cs.srv_block.wait_stats.bior_done	
exp.clf.unix.cs.srv_block.wait_stats.bior_time	
exp.clf.unix.cs.srv_block.wait_stats.bior_waits	
exp.clf.unix.cs.srv_block.wait_stats.biow_done	
exp.clf.unix.cs.srv_block.wait_stats.biow_time	
exp.clf.unix.cs.srv_block.wait_stats.biow_waits	
exp.clf.unix.cs.srv_block.wait_stats.dio_done	Total DIOs completed
exp.clf.unix.cs.srv_block.wait_stats.dio_idle	Total DIOs completed while idle
exp.clf.unix.cs.srv_block.wait_stats.dio_time	Tick count in DIO wait
exp.clf.unix.cs.srv_block.wait_stats.dio_waits	Number of waits for DIO
exp.clf.unix.cs.srv_block.wait_stats.dior_done	
exp.clf.unix.cs.srv_block.wait_stats.dior_time	
exp.clf.unix.cs.srv_block.wait_stats.dior_waits	
exp.clf.unix.cs.srv_block.wait_stats.diow_done	
exp.clf.unix.cs.srv_block.wait_stats.diow_time	
exp.clf.unix.cs.srv_block.wait_stats.diow_waits	
exp.clf.unix.cs.srv_block.wait_stats.event_count	
exp.clf.unix.cs.srv_block.wait_stats.event_wait	
exp.clf.unix.cs.srv_block.wait_stats.lg_done	Total LG completions
exp.clf.unix.cs.srv_block.wait_stats.lg_idle	LG completions while idle
exp.clf.unix.cs.srv_block.wait_stats.lg_time	Tick count in LG wait
exp.clf.unix.cs.srv_block.wait_stats.lg_waits	Number of waits for LG
exp.clf.unix.cs.srv_block.wait_stats.lge_done	
exp.clf.unix.cs.srv_block.wait_stats.lge_time	
exp.clf.unix.cs.srv_block.wait_stats.lge_waits	

exp.clf.unix.cs.srv_block.wait_stats.lio_done	
exp.clf.unix.cs.srv_block.wait_stats.lio_time	
exp.clf.unix.cs.srv_block.wait_stats.lio_waits	
exp.clf.unix.cs.srv_block.wait_stats.lior_done	
exp.clf.unix.cs.srv_block.wait_stats.lior_time	
exp.clf.unix.cs.srv_block.wait_stats.lior_waits	
exp.clf.unix.cs.srv_block.wait_stats.liow_done	
exp.clf.unix.cs.srv_block.wait_stats.liow_time	
exp.clf.unix.cs.srv_block.wait_stats.liow_waits	
exp.clf.unix.cs.srv_block.wait_stats.lk_done	Total LK completions
exp.clf.unix.cs.srv_block.wait_stats.lk_idle	LK completions while idle
exp.clf.unix.cs.srv_block.wait_stats.lk_time	Tick count in LK wait
exp.clf.unix.cs.srv_block.wait_stats.lk_waits	Number of waits for LK
exp.clf.unix.cs.srv_block.wait_stats.lke_done	
exp.clf.unix.cs.srv_block.wait_stats.lke_time	
exp.clf.unix.cs.srv_block.wait_stats.lke_waits	
exp.clf.unix.cs.srv_block.wait_stats.tm_done	
exp.clf.unix.cs.srv_block.wait_stats.tm_idle	
exp.clf.unix.cs.srv_block.wait_stats.tm_time	
exp.clf.unix.cs.srv_block.wait_stats.tm_waits	
exp.clf.unix.cs.srv_block.write	
exp.clf.unix.cs.srv_block.wt_list	List of SCBs waiting for an event
exp.clf.unix.me.num.bytes_used	Number of bytes of memory (possible shared) requested by this server
exp.clf.unix.me.num.free_pages	Number of times the MEfree_pages routine was called to free memory pages back to the O/S
exp.clf.unix.me.num.get_pages	Number of times the MEget_pages routine was called to request memory from the O/S
exp.clf.unix.me.num.pages_used	Number of memory 'pages' in use by this server
exp.clf.unix.me.num.test_count	DEBUGGING CONTROL OBJECT–tests internal memory routines–DO NOT USE
exp.clf.unix.me.num.test_errs	DEBUGGING CONTROL OBJECT–causes internal memory test routines to be

	executed–DO NOT USE
exp.clf.unix.me.num.test_mod	DEBUGGING CONTROL OBJECT–tests internal memory routines–DO NOT USE
exp.dmf.dm0p.bm_bufcnt	Number of buffers (single page and group) in the cache
exp.dmf.dm0p.bm_check	The number of times that a cached page was checked to see if it was still valid
exp.dmf.dm0p.bm_clock	
exp.dmf.dm0p.bm_dirty	Number of times a dirty page was unfixed (removed from cache)
exp.dmf.dm0p.bm_fcount	Number of free buffers
exp.dmf.dm0p.bm_fcwait	
exp.dmf.dm0p.bm_fix	Number of fix calls (fetching a page into cache is "fix"ing it)
exp.dmf.dm0p.bm_flimit	Lower limit on free list size (used to control forced-write behavior)
exp.dmf.dm0p.bm_force	Number of 'forces'–intermediate writes before the end of transaction.
exp.dmf.dm0p.bm_fwait	Number of free buffer waiters
exp.dmf.dm0p.bm_gcnt	Number of group buffers
exp.dmf.dm0p.bm_gfcount	Number of free group buffers
exp.dmf.dm0p.bm_glcount	Number of fixed group buffers
exp.dmf.dm0p.bm_gmcount	Number of modified group buffers
exp.dmf.dm0p.bm_gpages	Number of pages in a group buffer
exp.dmf.dm0p.bm_greads	Number of group reads performed
exp.dmf.dm0p.bm_gsyncwr	
exp.dmf.dm0p.bm_gwait	
exp.dmf.dm0p.bm_gwrites	Number of group writes performed
exp.dmf.dm0p.bm_hit	Number of times that a requested page is found in cache
exp.dmf.dm0p.bm_hshcnt	Number of hash buckets
exp.dmf.dm0p.bm_iowait	Number of stalls for i/o completion
exp.dmf.dm0p.bm_lcount	Number of fixed buffers
exp.dmf.dm0p.bm_mcount	Number of modified buffers
exp.dmf.dm0p.bm_mlimit	Upper limit on modify list size

exp.dmf.dm0p.bm_mwait	Number of mutex waits
exp.dmf.dm0p.bm_pgsize	
exp.dmf.dm0p.bm_reads	Number of single page reads
exp.dmf.dm0p.bm_reclaim	
exp.dmf.dm0p.bm_refresh	Number of times that a cached page was found to be invalid and was refreshed
exp.dmf.dm0p.bm_replace	
exp.dmf.dm0p.bm_sbufcnt	Number of single page buffers in the cache
exp.dmf.dm0p.bm_status	Buffer manager status: look up using ima_dmf_cache_status in IMA_STATUS_LOOKUP
exp.dmf.dm0p.bm_syncwr	
exp.dmf.dm0p.bm_unfix	Number of unfix calls–the number of times a page was released from cache.
exp.dmf.dm0p.bm_wbend	Number of pages on modify list at which to stop asynchronous write behind threads
exp.dmf.dm0p.bm_wbstart	Number of pages on modify list at which to start asynchronous write behind threads
exp.dmf.dm0p.bm_writes	Number of writes
exp.dmf.dm0p.bmc_bmcwait	
exp.dmf.dm0p.bmc_cpcheck	
exp.dmf.dm0p.bmc_cpcount	
exp.dmf.dm0p.bmc_cpindex	
exp.dmf.dm0p.bmc_dbcsize	
exp.dmf.dm0p.bmc_fcflush	
exp.dmf.dm0p.bmc_lockreclaim	
exp.dmf.dm0p.bmc_srv_count	
exp.dmf.dm0p.bmc_status	
exp.dmf.dm0p.bmc_tblcsize	
exp.dmf.dm0p.lbm_check	Number of times that a page in the local buffer manager was checked to make sure it was still valid
exp.dmf.dm0p.lbm_dirty	Dirty pages unfixed
exp.dmf.dm0p.lbm_fcwait	
exp.dmf.dm0p.lbm_fix	Number of fix calls

exp.dmf.dm0p.lbm_force	Immediate writes
exp.dmf.dm0p.lbm_fwait	Free buffer waiters
exp.dmf.dm0p.lbm_greads	Group reads
exp.dmf.dm0p.lbm_gsyncwr	
exp.dmf.dm0p.lbm_gwait	
exp.dmf.dm0p.lbm_gwrites	Group writes
exp.dmf.dm0p.lbm_hit	Page found is cache
exp.dmf.dm0p.lbm_iowait	Stall for I/O completion
exp.dmf.dm0p.lbm_mwait	Mutex waits
exp.dmf.dm0p.lbm_pgsize	
exp.dmf.dm0p.lbm_reads	Reads
exp.dmf.dm0p.lbm_reclaim	
exp.dmf.dm0p.lbm_refresh	Cached page was invalid
exp.dmf.dm0p.lbm_replace	
exp.dmf.dm0p.lbm_syncwr	
exp.dmf.dm0p.lbm_unfix	Unfix calls
exp.dmf.dm0p.lbm_wb_active	
exp.dmf.dm0p.lbm_wb_flush	
exp.dmf.dm0p.lbm_wb_flushed	
exp.dmf.dm0p.lbm_wb_gflushed	
exp.dmf.dm0p.lbm_wb_hwm	
exp.dmf.dm0p.lbm_wb_threads	
exp.dmf.dm0p.lbm_writes	Writes
exp.dmf.dm0p.lbmc_bmcwait	
exp.dmf.dm0p.lbmc_fcflush	
exp.dmf.dm0p.lbmc_lockreclaim	
exp.dmf.dm0p.wb_active	
exp.dmf.dm0p.wb_flush	
exp.dmf.dm0p.wb_flushed	
exp.dmf.dm0p.wb_gflushed	
exp.dmf.dm0p.wb_hwm	
exp.dmf.dm0p.wb_threads	

exp.dmf.lg.ldb_buffer	Database information
exp.dmf.lg.ldb_d_first_la	String of first log record (for dump) associated with this DB
exp.dmf.lg.ldb_d_first_la.la_block	
exp.dmf.lg.ldb_d_first_la.la_offset	Offset of first dump log record
exp.dmf.lg.ldb_d_first_la.la_sequence	Sequence of first dump log record
exp.dmf.lg.ldb_d_last_la	String of last log address (for dump) associated with this DB
exp.dmf.lg.ldb_d_last_la.la_block	
exp.dmf.lg.ldb_d_last_la.la_offset	Offset of last dump log record
exp.dmf.lg.ldb_d_last_la.la_sequence	Sequence of last dump log record
exp.dmf.lg.ldb_database_id	Database ID for this database according to the iidatabase catalog in iidbdb
exp.dmf.lg.ldb_db_name	The name of this database
exp.dmf.lg.ldb_db_owner	The username of the DBA of this database
exp.dmf.lg.ldb_id.id_id	The id_id value for the database open in the logging system. Each database has a unique id_id value and a recycle id_instance count. For the ID displayed in logstat or ipm, use 65536 * id_id + id_instance.
exp.dmf.lg.ldb_id.id_instance	Instance value of this db_id_id–for recycling
exp.dmf.lg.ldb_j_first_la	String of first journal log record
exp.dmf.lg.ldb_j_first_la.la_block	
exp.dmf.lg.ldb_j_first_la.la_offset	First journal log record offset
exp.dmf.lg.ldb_j_first_la.la_sequence	First journal log record sequence
exp.dmf.lg.ldb_j_last_la	String of last journal log record
exp.dmf.lg.ldb_j_last_la.la_block	
exp.dmf.lg.ldb_j_last_la.la_offset	Last journal log record offset
exp.dmf.lg.ldb_j_last_la.la_sequence	Last journal log record sequence
exp.dmf.lg.ldb_l_buffer	Number of characters in the ldb_buffer
exp.dmf.lg.ldb_lpd_count	Number of LPD references to this database
exp.dmf.lg.ldb_lxb_count	Number of transactions
exp.dmf.lg.ldb_lxbo_count	Number of ongoing transactions for checkpoint
exp.dmf.lg.ldb_sback_lsn	String of start of backup log sequence

	number
exp.dmf.lg.ldb_sback_lsn_high	High part of start backup LSN
exp.dmf.lg.ldb_sback_lsn_low	Low part of start backup LSN
exp.dmf.lg.ldb_sbackup	String of start backup LGA
exp.dmf.lg.ldb_sbackup.la_block	
exp.dmf.lg.ldb_sbackup.la_offset	Offset of start backup LGA
exp.dmf.lg.ldb_sbackup.la_sequence	Sequence of start backup LGA
exp.dmf.lg.ldb_stat.begin	Begins by database
exp.dmf.lg.ldb_stat.end	Ends by database
exp.dmf.lg.ldb_stat.force	Force by database
exp.dmf.lg.ldb_stat.read	Reads by database
exp.dmf.lg.ldb_stat.wait	Waits by database
exp.dmf.lg.ldb_stat.write	Writes by database
exp.dmf.lg.ldb_status	Status string for a database attached to the logging system.
exp.dmf.lg.ldb_status_num	Status as a bitmask–look up ima_db_status in IMA_STATUS_LOOKUP
exp.dmf.lg.lfb_active_log	String of the active log, either the primary or the dual
exp.dmf.lg.lfb_active_log_num	Active log number–look up ima_active_log_files in IMA_STATUS_LOOKUP
exp.dmf.lg.lfb_archive_end	
exp.dmf.lg.lfb_archive_end.la_block	
exp.dmf.lg.lfb_archive_end.la_offset	
exp.dmf.lg.lfb_archive_end.la_sequence	
exp.dmf.lg.lfb_archive_prevcp	
exp.dmf.lg.lfb_archive_prevcp.la_block	
exp.dmf.lg.lfb_archive_prevcp.la_offset	
exp.dmf.lg.lfb_archive_prevcp.la_sequence	
exp.dmf.lg.lfb_archive_start	
exp.dmf.lg.lfb_archive_start.la_block	
exp.dmf.lg.lfb_archive_start.la_offset	
exp.dmf.lg.lfb_archive_start.la_sequence	

exp.dmf.lg.lfb_buf_cnt	Count of log buffers
exp.dmf.lg.lfb_channel_blk	Last block written/read to/from the file associated with the main channel
exp.dmf.lg.lfb_dual_channel_blk	Last block written/read to/from the file associated with the dual channel
exp.dmf.lg.lfb_forced_lga	String of forced LGA
exp.dmf.lg.lfb_forced_lga.la_block	
exp.dmf.lg.lfb_forced_lga.la_offset	Offset of forced LGA
exp.dmf.lg.lfb_forced_lga.la_sequence	Sequence of forced LGA
exp.dmf.lg.lfb_forced_lsn	High part of forced LSN
exp.dmf.lg.lfb_forced_lsn_high	
exp.dmf.lg.lfb_forced_lsn_low	Low part of force LSN
exp.dmf.lg.lfb_hdr_lgh_active_logs	Active Log files–look up as ima_active_log_files in IMA_STATUS_LOOKUP
exp.dmf.lg.lfb_hdr_lgh_begin	String of log address of last begin in the log file
exp.dmf.lg.lfb_hdr_lgh_begin_blk	
exp.dmf.lg.lfb_hdr_lgh_begin_off	Offset of last begin
exp.dmf.lg.lfb_hdr_lgh_begin_seq	Sequence of last begin
exp.dmf.lg.lfb_hdr_lgh_checksum	Checksum of file header
exp.dmf.lg.lfb_hdr_lgh_count	Number of log pages
exp.dmf.lg.lfb_hdr_lgh_cp	String of log address of last consistency point
exp.dmf.lg.lfb_hdr_lgh_cp_blk	
exp.dmf.lg.lfb_hdr_lgh_cp_off	Offset of last CP
exp.dmf.lg.lfb_hdr_lgh_cp_seq	Sequence of last CP
exp.dmf.lg.lfb_hdr_lgh_cpcnt	Maximum CP interval for invoking archiver
exp.dmf.lg.lfb_hdr_lgh_end	String of log address of last end of file
exp.dmf.lg.lfb_hdr_lgh_end_blk	
exp.dmf.lg.lfb_hdr_lgh_end_off	Offset of last end of file
exp.dmf.lg.lfb_hdr_lgh_end_seq	Sequence of last end of file
exp.dmf.lg.lfb_hdr_lgh_l_abort	Force abort limit
exp.dmf.lg.lfb_hdr_lgh_l_cp	Expected next CP

exp.dmf.lg.lfb_hdr_lgh_l_logfull	Log Full limit
exp.dmf.lg.lfb_hdr_lgh_last_lsn	
exp.dmf.lg.lfb_hdr_lgh_last_lsn_lsn_high	
exp.dmf.lg.lfb_hdr_lgh_last_lsn_lsn_low	
exp.dmf.lg.lfb_hdr_lgh_percentage_logfull	
exp.dmf.lg.lfb_hdr_lgh_size	Size of a log page
exp.dmf.lg.lfb_hdr_lgh_status	String of log header status
exp.dmf.lg.lfb_hdr_lgh_status_num	Log Header Status–look up as ima_log_header_status in IMA_STATUS_LOOKUP
exp.dmf.lg.lfb_hdr_lgh_tran_high	High part of last xact id used
exp.dmf.lg.lfb_hdr_lgh_tran_id	String of last xact id used
exp.dmf.lg.lfb_hdr_lgh_tran_low	Low part of last xact id used
exp.dmf.lg.lfb_hdr_lgh_version	Logging system version–look up as ima_log_version in IMA_STATUS_LOOKUP
exp.dmf.lg.lfb_reserved_space	Amount of logfile space reserved
exp.dmf.lg.lfb_stat_dual_readio	Number of read from the II_DUAL_LOG
exp.dmf.lg.lfb_stat_dual_writeio	Number of write complete to the II_LOG_FILE
exp.dmf.lg.lfb_stat_end	Transactions ended on this logging system
exp.dmf.lg.lfb_stat_force	Log forces
exp.dmf.lg.lfb_stat_kbytes	Amount of (512 byte) blocks written
exp.dmf.lg.lfb_stat_log_readio	Number of read from the II_LOG_FILE
exp.dmf.lg.lfb_stat_log_writeio	Number of write complete to the II_LOG_FILE
exp.dmf.lg.lfb_stat_split	Number of log splits
exp.dmf.lg.lfb_stat_wait	Log waits
exp.dmf.lg.lfb_stat_write	Log writes
exp.dmf.lg.lfb_stat_writeio	Actual write I/Os to this log
exp.dmf.lg.lfb_status	String of status of this log file
exp.dmf.lg.lfb_status_num	Logging system status–look up as ima_log_file_status in IMA_STATUS_LOOKUP
exp.dmf.lg.lgd_check_stall	Point at which checking for various stall and logfull conditions (smallest of lgh_l_logfull,

	lgh_l_abort, lgd_cpstall) should begin
exp.dmf.lg.lgd_cnodeid	Node ID number for this local node. If not a cluster, this number will be 0.
exp.dmf.lg.lgd_cpstall	Maximum log file can grow to while executing a consistency point
exp.dmf.lg.lgd_csp_pid	The Process ID of the CSP
exp.dmf.lg.lgd_gcmt_numticks	Number of piggy-back write "ticks" after which a buffer has waited long enough and should be forced to disk.
exp.dmf.lg.lgd_gcmt_threshold	Threshold value at which group commit should be performed. As soon as there are more active protected transactions than this, group commit will begin.
exp.dmf.lg.lgd_ldb_inuse	
exp.dmf.lg.lgd_ldbb_count	
exp.dmf.lg.lgd_ldbb_size	
exp.dmf.lg.lgd_lfbb_count	
exp.dmf.lg.lgd_lfbb_size	
exp.dmf.lg.lgd_lpb_inuse	Number of LPBs in use
exp.dmf.lg.lgd_lpbb_count	
exp.dmf.lg.lgd_lpbb_size	
exp.dmf.lg.lgd_lpd_inuse	Number of LPDs in use
exp.dmf.lg.lgd_lpdb_count	
exp.dmf.lg.lgd_lpdb_size	
exp.dmf.lg.lgd_lxb_inuse	Number of LXBs in use
exp.dmf.lg.lgd_lxbb_count	
exp.dmf.lg.lgd_lxbb_size	
exp.dmf.lg.lgd_n_logwriters	Total number of available logwriter threads
exp.dmf.lg.lgd_no_bcp	CP in progress; do not start new one
exp.dmf.lg.lgd_protect_count	Number of protect transactions
exp.dmf.lg.lgd_stat.add	Log add calls
exp.dmf.lg.lgd_stat.bcp_stall_wait	Stalls caused by BCP writes
exp.dmf.lg.lgd_stat.begin	Log transaction begins
exp.dmf.lg.lgd_stat.dual_readio	Number of read from the II_DUAL_LOG
exp.dmf.lg.lgd_stat.dual_writeio	Number of write complete to the

	II_DUAL_LOG
exp.dmf.lg.lgd_stat.end	Log transaction end
exp.dmf.lg.lgd_stat.force	Log force call
exp.dmf.lg.lgd_stat.free_wait	Waits for free log buffer
exp.dmf.lg.lgd_stat.group_count	
exp.dmf.lg.lgd_stat.group_force	Number of group forces
exp.dmf.lg.lgd_stat.inconsist_db	Number of inconsistent databases occurred
exp.dmf.lg.lgd_stat.kbytes	Number of (512 byte) blocks written to the log file.
exp.dmf.lg.lgd_stat.log_readio	Number of read from the II_LOG_FILE
exp.dmf.lg.lgd_stat.log_writeio	Number of write complete to the II_LOG_FILE
exp.dmf.lg.lgd_stat.pgyback_check	Number of pgyback_write checks
exp.dmf.lg.lgd_stat.pgyback_write	Writes initiated by piggy-back
exp.dmf.lg.lgd_stat.readio	Log read I/Os
exp.dmf.lg.lgd_stat.remove	Log remove calls
exp.dmf.lg.lgd_stat.split	Log splits
exp.dmf.lg.lgd_stat.stall_wait	Number of times stalled in LGwrite
exp.dmf.lg.lgd_stat.wait	Log waits
exp.dmf.lg.lgd_stat.write	Log write calls
exp.dmf.lg.lgd_stat.writeio	Log write I/Os
exp.dmf.lg.lgd_status	String of logging system status
exp.dmf.lg.lgd_status_num	Integer of log status bits: look up ima_logging_system_status in IMA_STATUS_LOOKUP
exp.dmf.lg.lpb_bufmgr_id	ID of process' Buffer Manager
exp.dmf.lg.lpb_cond	Fast commit server condition look up ima_fcserver_cond in IMA_STATUS_LOOKUP
exp.dmf.lg.lpb_force_abort_sid	Session ID of this process' force abort special thread
exp.dmf.lg.lpb_gcmt_asleep	Indicator to track the state of the group commit thread 0–performing group commit;non zero–suspending
exp.dmf.lg.lpb_gcmt_lxb	
exp.dmf.lg.lpb_gcmt_sid	Session ID of this process' group commit

	special thread
exp.dmf.lg.lpb_id.id_id	The unique identifier associated with a process in the logging system. Each process has a unique id_id and a recycle id_instance counter. To get the id displayed in logstat, use (65536 * id_id) + id_instance
exp.dmf.lg.lpb_id.id_instance	Recycle counter for the id_id of this log process
exp.dmf.lg.lpb_lpd_count	Count of open databases
exp.dmf.lg.lpb_pid	Process id of the process attached to the logging system as a decimal number (for VMS, convert to hex)
exp.dmf.lg.lpb_stat.begin	Log begin per process
exp.dmf.lg.lpb_stat.end	Log end per process
exp.dmf.lg.lpb_stat.force	Log force per process
exp.dmf.lg.lpb_stat.readio	Log read I/Os per process
exp.dmf.lg.lpb_stat.wait	Log waits
exp.dmf.lg.lpb_stat.write	Log writes per process
exp.dmf.lg.lpb_status	String representing the status of the process
exp.dmf.lg.lpb_status_num	Status bits for this process–look up ima_process_status in IMA_STATUS_LOOKUP
exp.dmf.lg.lxb_cp_lga	String of CP interval for this transaction
exp.dmf.lg.lxb_cp_lga.la_block	
exp.dmf.lg.lxb_cp_lga.la_offset	Offset of CP interval for this transaction
exp.dmf.lg.lxb_cp_lga.la_sequence	Sequence of CP interval for this transaction
exp.dmf.lg.lxb_db_id_id	Id_id of the database associated with this transaction
exp.dmf.lg.lxb_db_name	Name of the database for this transaction
exp.dmf.lg.lxb_db_owner	DBA of the database associated with this transaction
exp.dmf.lg.lxb_dis_tran_id_hexdump	String of hexdump of the tran id, for use by XA_RECOVER
exp.dmf.lg.lxb_first_lga	String representing the first log address associated with this transaction
exp.dmf.lg.lxb_first_lga.la_block	
exp.dmf.lg.lxb_first_lga.la_offset	Offset of begin record

exp.dmf.lg.lxb_first_lga.la_sequence	Sequence of begin record
exp.dmf.lg.lxb_id.id_id	The unique id_id value for this transaction. Each transaction has a unique id_id value and a recycle id_instance counter. To get the value displayed in ipm and logstat, use 65536 * id_id + id_instance
exp.dmf.lg.lxb_id.id_instance	Sequence number for the id_id value of this transaction
exp.dmf.lg.lxb_is_prepared	Y if this transaction is prepared; otherwise N
exp.dmf.lg.lxb_is_xa_dis_tran_id	String of XA transaction ID, if relevant
exp.dmf.lg.lxb_last_lga	String representing the last log address associated with this transaction
exp.dmf.lg.lxb_last_lga.la_block	
exp.dmf.lg.lxb_last_lga.la_offset	Offset of last log record
exp.dmf.lg.lxb_last_lga.la_sequence	Sequence of last log record
exp.dmf.lg.lxb_last_lsn	String of last log record
exp.dmf.lg.lxb_last_lsn_high	High part of last log recore LSN
exp.dmf.lg.lxb_last_lsn_low	Low part of last log record LSN
exp.dmf.lg.lxb_pid	Process ID of the process (server) which started this transaction. The value is returned as a decimal integer. For VMS–convert to hex.
exp.dmf.lg.lxb_pr_id_id	ID_id of the process (server) which 'owns' this transaction.
exp.dmf.lg.lxb_reserved_space	Logfile space reserved by xact
exp.dmf.lg.lxb_sequence	Sequence number
exp.dmf.lg.lxb_sid	The session id in the server which owns the transaction
exp.dmf.lg.lxb_stat.force	Log force by transaction
exp.dmf.lg.lxb_stat.split	Log splits by transaction
exp.dmf.lg.lxb_stat.wait	Log wait by transaction
exp.dmf.lg.lxb_stat.write	Log writes by transaction
exp.dmf.lg.lxb_status	Status of this transaction
exp.dmf.lg.lxb_status_num	Status integer–look up ima_tx_status in IMA_STATUS_LOOKUP
exp.dmf.lg.lxb_tran_id	The transaction ID for this transaction

exp.dmf.lg.lxb_tran_id.db_high_tran	'High' part of the transaction ID
exp.dmf.lg.lxb_tran_id.db_low_tran	'Low' part of the transaction ID
exp.dmf.lg.lxb_user_name	Name of the user associated with this transaction
exp.dmf.lg.lxb_wait_reason	String representation of the reason that this transaction is waiting
exp.dmf.lg.lxb_wait_reason_num	Wait reason–look up ima_tx_wait_reason in IMA_STATUS_LOOKUP
exp.dmf.lk.lkb_attribute	String of attributes for a lock
exp.dmf.lk.lkb_attribute_num	Lock attribute–look up ima_lk_attribute in IMA_STATUS_LOOKUP
exp.dmf.lk.lkb_count	Count of physical requests
exp.dmf.lk.lkb_cx_req.cx_lock_id	
exp.dmf.lk.lkb_grant_mode	String representing the granted mode of this lock (S, IX, and so on)
exp.dmf.lk.lkb_grant_mode_num	Grant mode–look up ima_lk_mode in IMA_STATUS_LOOKUP
exp.dmf.lk.lkb_id.id_id	The unique id_id of a lock block. Each locked resource has a unique lock associated with it–these locks are held on a lock list. Each lock block has a unique id_id value and a recycle id_instance value
exp.dmf.lk.lkb_llb_id_id	The id_id of the locklist to which this lock belongs
exp.dmf.lk.lkb_request_mode	String representing the mode in which this lock has been requested
exp.dmf.lk.lkb_request_mode_num	Request mode–look up ima_lk_mode in IMA_STATUS_LOOKUP
exp.dmf.lk.lkb_rsb_id_id	The id_id of the resource with which this lock is associated
exp.dmf.lk.lkb_state	The state of the lock as a string (granted, waiting, and so on)
exp.dmf.lk.lkb_state_num	Lock state number–look up ima_lk_state in IMA_STATUS_LOOKUP
exp.dmf.lk.lkd_csid	The cluster ID of the current node
exp.dmf.lk.lkd_csp_id	The process ID of the CSP
exp.dmf.lk.lkd_lbk_count	Allocated LBK entries
exp.dmf.lk.lkd_lbk_size	Maximum LBK entries

exp.dmf.lk.lkd_lkb_inuse	Count of LKBs
exp.dmf.lk.lkd_lkh_size	Active LLB entries
exp.dmf.lk.lkd_llb_inuse	
exp.dmf.lk.lkd_lock_stall	Type of lock stall–look up ima_lock_stall_type in IMA_STATUS_LOOKUP
exp.dmf.lk.lkd_max_lkb	Default maximum LKB per lock list
exp.dmf.lk.lkd_next_llbname	
exp.dmf.lk.lkd_node_fail	Cluster node failure indication
exp.dmf.lk.lkd_rbk_count	Allocated RBK entries
exp.dmf.lk.lkd_rbk_size	Maximum RBK entries
exp.dmf.lk.lkd_rsb_inuse	Count of RSBs
exp.dmf.lk.lkd_rsh_size	Size of RSB hash table
exp.dmf.lk.lkd_sbk_count	Allocated SBK entries
exp.dmf.lk.lkd_sbk_size	Maximum SBK entries
exp.dmf.lk.lkd_stat.allocate_cb	Number of allocate_cb calls
exp.dmf.lk.lkd_stat.asynch_complete	Number of enq_complete completions
exp.dmf.lk.lkd_stat.cancel	Number of timeout cancels
exp.dmf.lk.lkd_stat.cnt_gdlck_call	Number of continue deadlock search calls
exp.dmf.lk.lkd_stat.cnt_gdlck_sent	Number of continue deadlock message sent
exp.dmf.lk.lkd_stat.convert	Number of explicit convert requests
exp.dmf.lk.lkd_stat.convert_deadlock	Number of converts that deadlocked
exp.dmf.lk.lkd_stat.convert_search	Number of convert deadlock searches
exp.dmf.lk.lkd_stat.convert_wait	Number of converts that waited
exp.dmf.lk.lkd_stat.create_list	Number of create list calls
exp.dmf.lk.lkd_stat.csp_msgs_rcvd	Number of messages received by CSP so far
exp.dmf.lk.lkd_stat.csp_wakeups_sent	Number of LK_wakeup calls made by CSP
exp.dmf.lk.lkd_stat.deadlock	Number of requests that deadlocked
exp.dmf.lk.lkd_stat.deadlock_search	Number of deadlock searches performed
exp.dmf.lk.lkd_stat.deallocate_cb	Number of deallocate_cb calls
exp.dmf.lk.lkd_stat.deq	Number of deq requested
exp.dmf.lk.lkd_stat.dlock_locks_examined	Number of LKBs seen by deadlock
exp.dmf.lk.lkd_stat.enq	Number of enq requested

exp.dmf.lk.lkd_stat.gdeadlock	Number of global deadlock detected
exp.dmf.lk.lkd_stat.gdlck_call	Number of continue deadlock search calls
exp.dmf.lk.lkd_stat.gdlck_grant	Number of global locks grant before the the global deadlock search
exp.dmf.lk.lkd_stat.gdlck_search	Number of pending global deadlock search requests
exp.dmf.lk.lkd_stat.gdlck_sent	Number of global deadlock messages sent
exp.dmf.lk.lkd_stat.lbk_highwater	Highwater mark of LBK allocation
exp.dmf.lk.lkd_stat.lkb_highwater	
exp.dmf.lk.lkd_stat.llb_highwater	Highwater mark of LLB usage
exp.dmf.lk.lkd_stat.max_lcl_dlk_srch	Max length of a deadlocksearch
exp.dmf.lk.lkd_stat.max_lock_chain_len	Max length of a LKB chain search
exp.dmf.lk.lkd_stat.max_locks_per_txn	
exp.dmf.lk.lkd_stat.max_rsrc_chain_len	Max length of a RSB chain search
exp.dmf.lk.lkd_stat.rbk_highwater	Highwater mark of RBK allocation
exp.dmf.lk.lkd_stat.release	Number of release physical locks
exp.dmf.lk.lkd_stat.release_all	Number of release lists
exp.dmf.lk.lkd_stat.release_partial	Number of partial releases
exp.dmf.lk.lkd_stat.request_convert	Number of request with implied conversions
exp.dmf.lk.lkd_stat.request_new	Number of request new lock calls
exp.dmf.lk.lkd_stat.rsb_highwater	Highwater mark of RSB usage
exp.dmf.lk.lkd_stat.sbk_highwater	Highwater mark of SBK allocation
exp.dmf.lk.lkd_stat.sent_all	Number of sent all deadlock search request
exp.dmf.lk.lkd_stat.synch_complete	Number of SS$_SYNCH completions
exp.dmf.lk.lkd_stat.totl_gdlck_search	Number of global deadlock search requests
exp.dmf.lk.lkd_stat.wait	Number of requests that waited
exp.dmf.lk.lkd_stat.walk_wait_for	
exp.dmf.lk.lkd_status	String of global locking system status.
exp.dmf.lk.lkd_status_num	Bitmask of global locking system status: look up ima_locking_system_status in IMA_STATUS_LOOKUP
exp.dmf.lk.llb_connect_count	Number of LLBs that reference this list through llb_shared_llb–used only if status LLB_SHARED

exp.dmf.lk.llb_event.type_high	
exp.dmf.lk.llb_event.type_low	
exp.dmf.lk.llb_event.value	
exp.dmf.lk.llb_evflags	Zero or LK_CROSS_PROCESS_EVENT(1)
exp.dmf.lk.llb_ew_stamp	
exp.dmf.lk.llb_id.id_id	Index for the lock list block–each locklist has an ID_ID (which is unique) and an ID_INSTANCE (which is a cycle count)–to get the value displayed in lockstat, use (65536 * id_id) + id_instance
exp.dmf.lk.llb_lkb_count	The count of locks on this list (each lock is known as a lock block or blk)
exp.dmf.lk.llb_llkb_count	The count of logical locks on this list
exp.dmf.lk.llb_max_lkb	The maximum locks permitted on this locklist
exp.dmf.lk.llb_name0	High byte of an internal unique lock list identifier (used in clusters)
exp.dmf.lk.llb_name1	Low byte of internal lock list identifier used in clusters
exp.dmf.lk.llb_pid	PID of server owning this lock list as a decimal number (convert to hex for VMS)
exp.dmf.lk.llb_related_count	The count of related LLBs
exp.dmf.lk.llb_related_llb	The related lock list, as a pointer
exp.dmf.lk.llb_related_llb_id_id	The id_id of the related lock list
exp.dmf.lk.llb_search_count	Deadlock searches on this list
exp.dmf.lk.llb_sid	Session ID of the session which owns this locklist (not unique in the installation, but only in a server)
exp.dmf.lk.llb_stamp	The stamp of the current lock request
exp.dmf.lk.llb_status	String showing the status of the lock list
exp.dmf.lk.llb_status_num	Bitmask of locklist status–look up ima_lklist_status in IMA_STATUS_LOOKUP
exp.dmf.lk.llb_tick	The current clock tick when making the glocal deadlock search request
exp.dmf.lk.llb_wait_id_id	The id_id of the locklist which holds the lock that the owner of this locklist is waiting for. This object only has a value when the current locklist is blocked.

exp.dmf.lk.rsb_convert_mode	String of the convert mode for the resource
exp.dmf.lk.rsb_convert_mode_num	Integer value of the convert mode for the resource–look up ima_lk_mode in IMA_STATUS_LOOKUP
exp.dmf.lk.rsb_grant_mode	A string representing the mode of a lock granted on this resource.
exp.dmf.lk.rsb_grant_mode_num	Integer value of the granted mode for the resource–look up ima_lk_mode in IMA_STATUS_LOOKUP
exp.dmf.lk.rsb_id.id_id	The id_id of a resource being locked. Each resource has a unique id_id value and a recycle id_instance counter.
exp.dmf.lk.rsb_invalid	Zero if valid, non-zero if valid
exp.dmf.lk.rsb_name	A string representing the name/key of the resource
exp.dmf.lk.rsb_name0	The type of the resource/lock combination–look up the value for the ima_lk_type column in the IMA_STATUS_LOOKUP table
exp.dmf.lk.rsb_name1	ID of the database associated with this resource–look up the value in the iidatabase catalog in the iidbdb
exp.dmf.lk.rsb_name2	Reltid of the table associated with this resource–look up in the iirelation catalog of the relevant database
exp.dmf.lk.rsb_name3	Reltidx of the table/index associated with this resource–look up in the iirelation catalog of the appropriate database
exp.dmf.lk.rsb_name4	Page number (if appropriate) of the resource
exp.dmf.lk.rsb_name5	More RSB information depending on RSB type
exp.dmf.lk.rsb_name6	More RSB information depending on RSB type
exp.dmf.lk.rsb_value0	Lower part of value associated with the resource
exp.dmf.lk.rsb_value1	High part of the value associated with the resource
exp.dmf.perf.tmperf_collect	Control object: Any write causes a snapshot of resource use to be taken
exp.dmf.perf.tmperf_cpu.TM_msecs	Milliseconds of CPU used in the server

exp.dmf.perf.tmperf_cpu.TM_secs	Seconds of CPU time used in the server
exp.dmf.perf.tmperf_dio	DIO count for the server
exp.dmf.perf.tmperf_idrss	Current resident set size for the server
exp.dmf.perf.tmperf_majflt	Major faults for the server
exp.dmf.perf.tmperf_maxrss	Max resident set size for the server
exp.dmf.perf.tmperf_minflt	Minor faults for the server
exp.dmf.perf.tmperf_msgrcv	Messages received by the server
exp.dmf.perf.tmperf_msgsnd	Messages sent by the server
exp.dmf.perf.tmperf_msgtotal	Total messages sent/received by the server
exp.dmf.perf.tmperf_nivcsw	Involuntary context switches (time sliced out)
exp.dmf.perf.tmperf_nsignals	Number of signals received
exp.dmf.perf.tmperf_nswap	Number of swaps
exp.dmf.perf.tmperf_nvcsw	Number of voluntary context switches
exp.dmf.perf.tmperf_reads	Number of reads
exp.dmf.perf.tmperf_stime.TM_msecs	Milliseconds of system time
exp.dmf.perf.tmperf_stime.TM_secs	Seconds of system time
exp.dmf.perf.tmperf_utime.TM_msecs	Milliseconds of user time
exp.dmf.perf.tmperf_utime.TM_secs	Seconds of user time
exp.dmf.perf.tmperf_writes	Number of writes
exp.gcf.gca.assoc	Index of GCA association for the process, starting from 1–each connection from a client program is known as an association
exp.gcf.gca.assoc.client_id	
exp.gcf.gca.assoc.flags	Flags for the association
exp.gcf.gca.assoc.inbound	
exp.gcf.gca.assoc.partner_id	
exp.gcf.gca.assoc.partner_protocol	Integer value of the protocol used for the connection to the local partner
exp.gcf.gca.assoc.session_protocol	Integer value of the protocol for this connected session
exp.gcf.gca.assoc.userid	User ID associated with this association
exp.gcf.gca.assoc_count	
exp.gcf.gca.client	

exp.gcf.gca.client.api_version	
exp.gcf.gca.client.flags	
exp.gcf.gca.client.listen_address	
exp.gcf.gca.client.local_protocol	
exp.gcf.gca.client.server_class	
exp.gcf.gca.data_in	
exp.gcf.gca.data_out	
exp.gcf.gca.install_id	
exp.gcf.gca.msgs_in	
exp.gcf.gca.msgs_out	
exp.gcf.gca.trace_level	Current GCA trace level
exp.gcf.gca.trace_log	
exp.gcf.gcc.conn	
exp.gcf.gcc.conn.al_flags	
exp.gcf.gcc.conn.al_proto_lvl	
exp.gcf.gcc.conn.flags	
exp.gcf.gcc.conn.gca_assoc_id	
exp.gcf.gcc.conn.inbound	
exp.gcf.gcc.conn.lcl_addr.node	
exp.gcf.gcc.conn.lcl_addr.port	
exp.gcf.gcc.conn.lcl_addr.protocol	
exp.gcf.gcc.conn.pl_flags	
exp.gcf.gcc.conn.pl_proto_lvl	
exp.gcf.gcc.conn.rmt_addr.node	
exp.gcf.gcc.conn.rmt_addr.port	
exp.gcf.gcc.conn.rmt_addr.protocol	
exp.gcf.gcc.conn.sl_flags	
exp.gcf.gcc.conn.sl_proto_lvl	
exp.gcf.gcc.conn.target	
exp.gcf.gcc.conn.tl_flags	
exp.gcf.gcc.conn.tl_lcl_id	
exp.gcf.gcc.conn.tl_proto_lvl	

exp.gcf.gcc.conn.tl_rmt_id
exp.gcf.gcc.conn.trg_addr.node
exp.gcf.gcc.conn.trg_addr.port
exp.gcf.gcc.conn.trg_addr.protocol
exp.gcf.gcc.conn.userid
exp.gcf.gcc.conn_count
exp.gcf.gcc.data_in
exp.gcf.gcc.data_out
exp.gcf.gcc.ib_conn_count
exp.gcf.gcc.ib_encrypt_mech
exp.gcf.gcc.ib_encrypt_mode
exp.gcf.gcc.ib_max
exp.gcf.gcc.msgs_in
exp.gcf.gcc.msgs_out
exp.gcf.gcc.ob_conn_count
exp.gcf.gcc.ob_encrypt_mech
exp.gcf.gcc.ob_encrypt_mode
exp.gcf.gcc.ob_max
exp.gcf.gcc.pl_proto_lvl
exp.gcf.gcc.protocol
exp.gcf.gcc.protocol.addr
exp.gcf.gcc.protocol.host
exp.gcf.gcc.protocol.port
exp.gcf.gcc.registry
exp.gcf.gcc.registry.addr
exp.gcf.gcc.registry.host
exp.gcf.gcc.registry.port
exp.gcf.gcc.registry_mode
exp.gcf.gcc.trace_level
exp.gcf.gcn.bedcheck_interval
exp.gcf.gcn.cache_modified
exp.gcf.gcn.compress_point

exp.gcf.gcn.default_class	
exp.gcf.gcn.expire_interval	
exp.gcf.gcn.hostname	
exp.gcf.gcn.installation_id	
exp.gcf.gcn.local_vnode	
exp.gcf.gcn.remote_mechanism	
exp.gcf.gcn.remote_vnode	
exp.gcf.gcn.server.address	The GCA address of a registered server process, as seen by the name server (iigcn)
exp.gcf.gcn.server.class	The 'class' of the server registered with the name server (iigcn)–this corresponds to the server class parameter specified in CBF
exp.gcf.gcn.server.entry	
exp.gcf.gcn.server.flags	
exp.gcf.gcn.server.object	The 'dblist' of the DBMS server registered with the name server–relevant for DBMS servers only
exp.gcf.gcn.ticket_cache_size	
exp.gcf.gcn.ticket_expire	
exp.gcf.gcn.ticket_lcl_cache_miss	
exp.gcf.gcn.ticket_lcl_created	
exp.gcf.gcn.ticket_lcl_expired	
exp.gcf.gcn.ticket_lcl_used	
exp.gcf.gcn.ticket_rmt_cache_miss	
exp.gcf.gcn.ticket_rmt_created	
exp.gcf.gcn.ticket_rmt_expired	
exp.gcf.gcn.ticket_rmt_used	
exp.gcf.gcn.timeout	
exp.gcf.gcn.trace_level	
exp.gcf.gcs.default_mechanism	
exp.gcf.gcs.installation_mechanism	
exp.gcf.gcs.mechanism	
exp.gcf.gcs.mechanism.capabilities	
exp.gcf.gcs.mechanism.encrypt_level	

exp.gcf.gcs.mechanism.name	
exp.gcf.gcs.mechanism.overhead	
exp.gcf.gcs.mechanism.status	
exp.gcf.gcs.mechanism.version	
exp.gcf.gcs.trace_level	
exp.gcf.gcs.version	
exp.glf.mo.mem.bytes	
exp.glf.mo.mem.limit	
exp.glf.mo.meta.class	The textual class of the classid
exp.glf.mo.meta.classid	The classid key
exp.glf.mo.meta.index	The classid that indexes this classid, so this can be a column in a the
exp.glf.mo.meta.offset	The offset to hand to methods
exp.glf.mo.meta.oid	The ASN.1 OID to assign to this classid
exp.glf.mo.meta.perms	MO permissions for the object
exp.glf.mo.meta.size	MO size of the object
exp.glf.mo.monitors.classid	
exp.glf.mo.monitors.mon_block	
exp.glf.mo.monitors.mon_data	
exp.glf.mo.monitors.mon_id	
exp.glf.mo.monitors.monitor	
exp.glf.mo.monitors.qual_regexp	
exp.glf.mo.num.alloc	
exp.glf.mo.num.attach	
exp.glf.mo.num.classdef	
exp.glf.mo.num.del_monitor	
exp.glf.mo.num.detach	
exp.glf.mo.num.free	
exp.glf.mo.num.get	
exp.glf.mo.num.getnext	
exp.glf.mo.num.mutex	
exp.glf.mo.num.set_monitor	

exp.glf.mo.num.tell_monitor
exp.glf.mo.num.unmutex
exp.glf.mo.oid_map.file_name
exp.glf.mo.oid_map.file_time
exp.glf.mo.sptrees.enqcmps
exp.glf.mo.sptrees.enqs
exp.glf.mo.sptrees.lkpcmps
exp.glf.mo.sptrees.lookups
exp.glf.mo.sptrees.name
exp.glf.mo.sptrees.prevnexts
exp.glf.mo.sptrees.splayloops
exp.glf.mo.sptrees.splays
exp.glf.mo.strings.bytes
exp.glf.mo.strings.defined
exp.glf.mo.strings.deleted
exp.glf.mo.strings.freed
exp.glf.mo.strings.limit
exp.glf.mo.strings.refs
exp.glf.mo.strings.vals
exp.gwf.gwm.glb.aborts
exp.gwf.gwm.glb.begins
exp.gwf.gwm.glb.closes
exp.gwf.gwm.glb.cnd.connections
exp.gwf.gwm.glb.commits
exp.gwf.gwm.glb.connects.active
exp.gwf.gwm.glb.connects.max
exp.gwf.gwm.glb.connects.place.max
exp.gwf.gwm.glb.def_domain
exp.gwf.gwm.glb.def_vnode
exp.gwf.gwm.glb.deletes
exp.gwf.gwm.glb.errors
exp.gwf.gwm.glb.gcm.errs

exp.gwf.gwm.glb.gcm.readahead
exp.gwf.gwm.glb.gcm.reissues
exp.gwf.gwm.glb.gcm.sends
exp.gwf.gwm.glb.gcm.usecache
exp.gwf.gwm.glb.gcn_checks
exp.gwf.gwm.glb.gcn_queries
exp.gwf.gwm.glb.gets
exp.gwf.gwm.glb.idxfs
exp.gwf.gwm.glb.infos
exp.gwf.gwm.glb.inits
exp.gwf.gwm.glb.num.rop_calls
exp.gwf.gwm.glb.num.rop_hits
exp.gwf.gwm.glb.num.rop_misses
exp.gwf.gwm.glb.num.rreq_calls
exp.gwf.gwm.glb.num.rreq_hits
exp.gwf.gwm.glb.num.rreq_misses
exp.gwf.gwm.glb.opens
exp.gwf.gwm.glb.positions
exp.gwf.gwm.glb.puts
exp.gwf.gwm.glb.replaces
exp.gwf.gwm.glb.sem.places
exp.gwf.gwm.glb.sem.stat
exp.gwf.gwm.glb.tabfs
exp.gwf.gwm.glb.terms
exp.gwf.gwm.glb.this_server
exp.gwf.gwm.glb.time_to_live
exp.gwf.gwm.num.mob_aborts
exp.gwf.gwm.num.mob_bad_cpositions
exp.gwf.gwm.num.mob_bad_ipositions
exp.gwf.gwm.num.mob_begins
exp.gwf.gwm.num.mob_closes
exp.gwf.gwm.num.mob_commits

exp.gwf.gwm.num.mob_deletes
exp.gwf.gwm.num.mob_gets
exp.gwf.gwm.num.mob_idxfs
exp.gwf.gwm.num.mob_inits
exp.gwf.gwm.num.mob_opens
exp.gwf.gwm.num.mob_positions
exp.gwf.gwm.num.mob_puts
exp.gwf.gwm.num.mob_replaces
exp.gwf.gwm.num.mob_tabfs
exp.gwf.gwm.num.mob_terms
exp.gwf.gwm.num.xt_aborts
exp.gwf.gwm.num.xt_bad_positions
exp.gwf.gwm.num.xt_begins
exp.gwf.gwm.num.xt_closes
exp.gwf.gwm.num.xt_commits
exp.gwf.gwm.num.xt_deletes
exp.gwf.gwm.num.xt_gets
exp.gwf.gwm.num.xt_idxfs
exp.gwf.gwm.num.xt_opens
exp.gwf.gwm.num.xt_positions
exp.gwf.gwm.num.xt_puts
exp.gwf.gwm.num.xt_replaces
exp.gwf.gwm.num.xt_tabfs
exp.gwf.gwm.num.xt_terms
exp.gwf.gwm.places.index
exp.gwf.gwm.places.sem
exp.gwf.gwm.places.type
exp.gwf.gwm.servers.class
exp.gwf.gwm.servers.cnd
exp.gwf.gwm.servers.conn_tree
exp.gwf.gwm.servers.connects
exp.gwf.gwm.servers.flags

exp.gwf.gwm.servers.flags_str	
exp.gwf.gwm.servers.state	
exp.gwf.gwm.servers.valid	
exp.gwf.gwm.session.control.add_vnode	Control object, valid only in SQL session. Write of a vnode or server address adds it to the session domain.
exp.gwf.gwm.session.control.del_vnode	Control object, valid only in SQL session. Write of vnode or server address deletes it from the session domain.
exp.gwf.gwm.session.control.reset_domain	Control object, valid only in SQL session. Any write resets the session domain to the default server.
exp.gwf.gwm.session.dom.index	
exp.gwf.gwm.trace.aborts	
exp.gwf.gwm.trace.begins	
exp.gwf.gwm.trace.closes	
exp.gwf.gwm.trace.commits	
exp.gwf.gwm.trace.deletes	
exp.gwf.gwm.trace.errors	
exp.gwf.gwm.trace.gets	
exp.gwf.gwm.trace.idxfs	
exp.gwf.gwm.trace.infos	
exp.gwf.gwm.trace.inits	
exp.gwf.gwm.trace.opens	
exp.gwf.gwm.trace.positions	
exp.gwf.gwm.trace.puts	
exp.gwf.gwm.trace.replaces	
exp.gwf.gwm.trace.tabfs	
exp.gwf.gwm.trace.terms	
exp.gwf.gwm.vnode.expire	
exp.gwf.sxa.close_count	
exp.gwf.sxa.close_fail_count	
exp.gwf.sxa.get_bytid	
exp.gwf.sxa.get_bytid_rescan	

exp.gwf.sxa.get_count	
exp.gwf.sxa.get_fail_count	
exp.gwf.sxa.key_position_tries	
exp.gwf.sxa.msgid_count	
exp.gwf.sxa.msgid_fail_count	
exp.gwf.sxa.msgid_lookup_count	
exp.gwf.sxa.open_count	
exp.gwf.sxa.open_fail_count	
exp.gwf.sxa.reg_fail_count	
exp.gwf.sxa.reg_index_try_count	
exp.gwf.sxa.register_count	
exp.gwf.sxa.update_attempts	
exp.qsf.qso.dbp.dusage	
exp.qsf.qso.dbp.index	Unique index value representing a database procedure in the QSF pool
exp.qsf.qso.dbp.lstate	
exp.qsf.qso.dbp.name	Human readable name of the database procedure
exp.qsf.qso.dbp.owner	Login name of the user who owns (created) this database procedure
exp.qsf.qso.dbp.size	Size in bytes of this database procedure
exp.qsf.qso.dbp.udbid	ID of the database in which this database procedure was created (look up the id column in iidatabase)
exp.qsf.qso.dbp.usage	Usage counter for this database procedure
exp.qsf.qso.rqp.index	Unique index value representing a repeated query in the QSF pool
exp.qsf.qso.rqp.name	Human readable form of the name of this repeated query. The numeric values are the values used by the application when the query is defined.
exp.qsf.qso.rqp.size	Size in bytes of this repeated query
exp.qsf.qso.rqp.udbid	ID of the database in which this repeat query is defined. Look up the value of the id column in iidatabase
exp.qsf.qso.rqp.usage	Usage counter for this repeat query

exp.qsf.qsr.qsr_bkts_used	Number of hash buckets that actually have at least one object in them (QSF uses a hashing algorithm to speed up access to objects in the cache)
exp.qsf.qsr.qsr_bmaxobjs	Max number of objects ever in any one given bucket
exp.qsf.qsr.qsr_decay_factor	
exp.qsf.qsr.qsr_memleft	Free memory within the QSF pool
exp.qsf.qsr.qsr_memtot	Total amount of memory to use for QSF's memory pool.
exp.qsf.qsr.qsr_mx_index	High watermark (maximum) number of indexed objects
exp.qsf.qsr.qsr_mx_named	High watermark (maximum) of Named Objects
exp.qsf.qsr.qsr_mx_rsize	Size of the largest single object requested from the QSF pool
exp.qsf.qsr.qsr_mx_size	Size of the largets object stored in the QSF pool
exp.qsf.qsr.qsr_mx_unnamed	High watermark of the number of unnamed objects stored in this QSF pool
exp.qsf.qsr.qsr_mxbkts_used	High watermark of the number of hash buckets that have been used in this QSF pool
exp.qsf.qsr.qsr_mxobjs	High watermark of the number of objects stored in this QSF pool
exp.qsf.qsr.qsr_mxsess	High watermark of the number of concurrent active sessions accessing this QSF pool
exp.qsf.qsr.qsr_named_requests	Number of times that a named object was requested (since the last time the pool was flushed)
exp.qsf.qsr.qsr_nbuckets	Number of hash buckets in this QSF pool
exp.qsf.qsr.qsr_no_destroyed	Number of 'master' objects destroyed
exp.qsf.qsr.qsr_no_index	Current number of 'master' or indexed objects
exp.qsf.qsr.qsr_no_named	Current number of named objects in this QSF pool
exp.qsf.qsr.qsr_no_unnamed	Current number of unnamed objects
exp.qsf.qsr.qsr_nobjs	Number objects currently stored in QSF

exp.qsf.qsr.qsr_nsess	QSF sessions currently active
exp.rdf.rdi.cache.max_defaults	Maximum number of DEFAULTS that can be cached by RDF
exp.rdf.rdi.cache.max_ldb_descs	Maximum number of LDBs (databases) that can be cached by RDF
exp.rdf.rdi.cache.max_qtree_objs	Maximum number of Query Tree objects that can be cached by RDF
exp.rdf.rdi.cache.max_tables	Maximum number of tables whose information can be cached by RDF
exp.rdf.rdi.cache.size	Size of the RDF cache in bytes
exp.rdf.rdi.flags.cost_set	Indicates whether CPU/net cost are set for distributed query optimization
exp.rdf.rdi.num.bad_cksums	Counter for number of bad checksums detected
exp.rdf.rdi.num.cluster_nodes	Number of nodes known to cluster Ingres installation. 0 if not a cluster installation
exp.rdf.rdi.num.ddbs_caches	Number of distributed databases to cache
exp.rdf.rdi.num.max_sessions	Maximum number of sessions allowed in a server
exp.rdf.rdi.num.memleft	Remaining memory in the RDF pool, in bytes
exp.rdf.rdi.startup.avg_ldbs	Saved input parameter for the average number of ldbs per ddb
exp.rdf.rdi.startup.cache_ddbs	Saved input parameter for the maximum number of ddbs to cache
exp.rdf.rdi.startup.max_tables	Saved input parameter for the number of tables in the relation cache
exp.rdf.rdi.startup.memory	Saved input parameter for the RDF pool size, in bytes
exp.rdf.rdi.startup.table_columns	Saved input parameter for estimated average number of columns per table
exp.rdf.rdi.startup.table_synonyms	Saved input parameter for average number of synonyms per table
exp.rdf.rdi.state_num	Integer value representing the state of the RDF cache–look up ima_rdf_cache_status in the IMA_STATUS_LOOKUP
exp.rdf.rdi.state_str	State of the RDF cache as a human-readable string

exp.rdf.stat.a10_dbdb	Counter for the number of times a tuple has been appended to the iidbdb catalog
exp.rdf.stat.a16_syn	Counter for the number of times a tuple has been appended to the iisynomyms catalog
exp.rdf.stat.a17_comment	Counter for the number of times a tuple has been appended to the iicomment catalog
exp.rdf.stat.a21_seqs	
exp.rdf.stat.a2_permit	Counter for the number of times permission info has been appended to the catalogs
exp.rdf.stat.a3_integ	Counter for the number of times integrity info has been appended to the catalogs
exp.rdf.stat.a4_view	Counter for the number of times view info has been appended to the catalogs
exp.rdf.stat.a5_proc	Counter for the number of times procedure info has been appended to the catalogs
exp.rdf.stat.a6_rule	Counter for the number of times rule info has been appended to the catalogs
exp.rdf.stat.a7_event	Counter for the number of times event info has been appended to the catalogs
exp.rdf.stat.a8_alltoall	Counter for the number of times "all to public" permission has been set on an object
exp.rdf.stat.a9_rtoall	Counter for the number of times "retrieve to all"/"select to public" permission has been set on an object
exp.rdf.stat.b0_exists	
exp.rdf.stat.b11_stat	
exp.rdf.stat.b12_itups	Counter for the number of times integrity tuples were built and placed in the cache
exp.rdf.stat.b13_etups	Counter for the number of times requested event permits were obtained on single RDF call (these are never cached)
exp.rdf.stat.b14_ptups	Counter for the number of times permit tuples were built and placed on the cache
exp.rdf.stat.b15_rtups	Counter for the number of times rule tuples were built and placed on the cache
exp.rdf.stat.b16_syn	Counter for the number of times a tuple has been found in the iisynonyms catalog and placed on the cache
exp.rdf.stat.b18_pptups	Counter for the number of times procedure

	parameter tuples were built and placed on the cache
exp.rdf.stat.b19_ktups	Counter for the number of times iikey tuples were built and placed on cache
exp.rdf.stat.b1_core	
exp.rdf.stat.b21_seqs	
exp.rdf.stat.b2_permit	Counter for the number of times permission info has been built and placed on the cache
exp.rdf.stat.b3_integ	Counter for the number of times integrity info has been built and placed on the cache
exp.rdf.stat.b4_view	Counter for the number of times view info has been built and placed on the cache
exp.rdf.stat.b5_proc	Counter for the number of times procedure info has been built and placed on the cache
exp.rdf.stat.b6_rule	Counter for the number of times rule info has been built and placed on the cache
exp.rdf.stat.b7_event	Counter for the number of times event info has been built (this is never cached)
exp.rdf.stat.c0_exists	
exp.rdf.stat.c11_stat	
exp.rdf.stat.c12_itups	Counter for the number of times requested integrity tuples were already on cache
exp.rdf.stat.c14_ptups	Counter for the number of times requested permit tuples were already on the cache
exp.rdf.stat.c15_rtups	Counter for the number of times requested rule tuples were already on cache
exp.rdf.stat.c18_pptups	Counter for the number of times requested procedure parameter tuples were already on cache
exp.rdf.stat.c19_ktups	Counter for the number of times iikey tuples requested were already on the cache
exp.rdf.stat.c1_core	
exp.rdf.stat.c2_permit	Counter for the number of times requested permission info was already on the cache
exp.rdf.stat.c3_integ	Counter for the number of times requested integrity info was already on the cache
exp.rdf.stat.c4_view	Counter for the number of times requested view info was already on the cache

exp.rdf.stat.c5_proc	Counter for the number of times requested procedure info was already on the cache
exp.rdf.stat.c6_rule	Counter for the number of times requested rule info was already on the cache
exp.rdf.stat.d10_dbdb	Counter for the number of times a tuple has been deleted from the iidbdb catalog
exp.rdf.stat.d16_syn	Counter for the number of times a tuple has been deleted from the iisynonyms catalog
exp.rdf.stat.d17_comment	Counter for the number of times a tuple has been deleted from the iicomment catalog
exp.rdf.stat.d21_seqs	
exp.rdf.stat.d2_permit	Counter for the number of times permission info has been deleted from the catalogs
exp.rdf.stat.d3_integ	Counter for the number of times integrity info has been deleted from the catalogs
exp.rdf.stat.d4_view	Counter for the number of times view info has been deleted from the catalogs
exp.rdf.stat.d5_proc	Counter for the number of times procedure info has been deleted from the catalogs
exp.rdf.stat.d6_rule	Counter for the number of times rule info has been deleted from the catalogs
exp.rdf.stat.d7_event	Counter for the number of times event info has been deleted from the catalogs
exp.rdf.stat.d8_alltoall	Counter for the number of times "all to public" permission has been renamed from an object
exp.rdf.stat.d9_rtoall	Counter for the number of times "retrieve to all"/"select to public" permission has been removed from an object
exp.rdf.stat.f0_relation	Counter for the number of times a relation object is fixed in memory
exp.rdf.stat.f1_integrity	Counter for the number of times an integrity object is fixed in memory
exp.rdf.stat.f2_procedure	Counter for the number of times a procedure object is fixed in memory
exp.rdf.stat.f3_protect	Counter for the number of times a protect object is fixed in memory
exp.rdf.stat.f4_rule	Counter for the number of times a rule object is fixed in memory

exp.rdf.stat.i1_obj_invalid	Counter for the number of times a single object in the RDF cache is invalidated.
exp.rdf.stat.i2_p_invalid	Counter for the number of times a procedure object is invalidated from RDF cache
exp.rdf.stat.i3_flushed	Counter for the number of times an object to be invalidated is already flushed off of the cache–indicator of extreme concurrency conditions
exp.rdf.stat.i4_class_invalid	Counter for the number of times all protect or integrity or rule trees of this class have been invalidated
exp.rdf.stat.i5_tree_invalid	Counter for the number of times a protect/integrity/rule tree has been invalidated
exp.rdf.stat.i6_qtcache_invalid	Counter for the number of times the RDF Query Tree cache is invalidated
exp.rdf.stat.i7_cache_invalid	Counter for the number of times the whole RDF cache is invalidated
exp.rdf.stat.i8_cache_invalid	Counter for the number of times the ldbdesc cache is invalidated
exp.rdf.stat.l0_private	Counter for the number of times RDF makes a private RDR_INFO object because another thread has an update lock on the object that this thread wants to update. This is an indicator of RDF concurrency.
exp.rdf.stat.l1_private	Overflow counter for rds_f1_private
exp.rdf.stat.l2_supdate	Counter for the number of times a thread updates a shared RDR_INFO object.
exp.rdf.stat.l3_supdate	Overflow counter for rds_f2_supdate
exp.rdf.stat.m12_itups	Counter for the number of times requested integrity tuples required multiple reads
exp.rdf.stat.m13_etups	Counter for the number of times event permit tuples required multiple RDF calls
exp.rdf.stat.m14_ptups	Counter for the number of times permit tuples required multiple reads
exp.rdf.stat.m15_rtups	Counter for the number of times rule tuples required multiple reads
exp.rdf.stat.m18_pptups	Counter for the number of times procedure parameter tuples required multiple reads

exp.rdf.stat.m19_ktups	Counter for the number of times iikey tuples required multiple reads
exp.rdf.stat.n0_tables	Counter for the number of tables on the cache. This will include multiple instances of the same table. If a table is cleared from cache, then a cache entry for it is rebuilt. It will count twice. Used to calculate the average number of columns
exp.rdf.stat.n10_nomem	Counter for the number of times RDF runs out of memory on the cache and is unable to reclaim it from either the query tree or the relation cache. If this count is non-zero, user MUST tune RDF cache to increase memory size.
exp.rdf.stat.n11_LDBmem_claim	Counter for the number of times RDF runs out of memory on the cache and reclaims it from the LDBdesc cache
exp.rdf.stat.n1_avg_col	
exp.rdf.stat.n2_avg_idx	
exp.rdf.stat.n7_exceptions	Counter for the number of exceptions RDF has encountered
exp.rdf.stat.n8_QTmem_claim	Counter for the number of times RDF runs out of memory on the cache and reclaims it from the Query Tree cache
exp.rdf.stat.n9_relmem_claim	Counter for the number of times RDF runs out of memory on the cache and reclaims it from the Relation cache
exp.rdf.stat.o10_dbdb	Counter for the number of times a tuple has been replaced/purged from the iidbdb catalog
exp.rdf.stat.r0_exists	
exp.rdf.stat.r11_stat	
exp.rdf.stat.r12_itups	Counter for the number of times integrity tuples were requested
exp.rdf.stat.r13_etups	Counter for the number of times event permit tuples were requested
exp.rdf.stat.r14_ptups	Counter for the number of times permit tuples were requested
exp.rdf.stat.r15_rtups	Counter for the number of times rule tuples were requested

exp.rdf.stat.r16_syn	Counter for the number of times RDF has looked for a synonym
exp.rdf.stat.r18_pptups	Counter for the number of times procedure parameter tuples were requested
exp.rdf.stat.r19_ktups	Counter for the number of times iikey tuples were requested
exp.rdf.stat.r1_core	
exp.rdf.stat.r21_seqs	
exp.rdf.stat.r2_permit	Counter for the number of times permission info has been requested from RDF
exp.rdf.stat.r3_integ	Counter for the number of times integrity info has been requested from RDF
exp.rdf.stat.r4_view	Counter for the number of times view info has been requested from RDF
exp.rdf.stat.r5_proc	Counter for the number of times procedure info has been requested from RDF
exp.rdf.stat.r6_rule	Counter for the number of times rule info has been requested from RDF
exp.rdf.stat.r7_event	Counter for the number of times event info has been requested from RDF
exp.rdf.stat.rdf_n12_default_claim	Counter for the number of times RDF runs out of memory on the cache and reclaims it from the defaults cache
exp.rdf.stat.u0_protect	Counter for the number of times a shared protect object is unfixed
exp.rdf.stat.u10_private	Counter for the number of times private RDR_INFO is unfixed
exp.rdf.stat.u11_shared	Counter for the number of times shared RDR_INFO is unfixed
exp.rdf.stat.u1_integrity	Counter for the number of times a shared integrity object is unfixed
exp.rdf.stat.u2_procedure	Counter for the number of times a shared procedure object is unfixed
exp.rdf.stat.u3_rule	Counter for the number of times a shared rule object is unfixed
exp.rdf.stat.u4p_protect	Counter for the number of times a private protect object is unfixed

exp.rdf.stat.u5p_integrity	Counter for the number of times a private integrity object is unfixed
exp.rdf.stat.u6p_procedure	Counter for the number of times a private procedure object is unfixed
exp.rdf.stat.u7p_rule	Counter for the number of times a private rule object is unfixed
exp.rdf.stat.u8_invalid	Counter for the number of times an invalid qrymod object unfix has been requested. This should always be zero in a healthy environment.
exp.rdf.stat.u9_fail	Counter for the number of times a RDR_INFO unfix request failed. This should always be zero in a healthy environment.
exp.rdf.stat.x1_relation	Counter for the number of times a relation infoblk is refreshed on behalf of caller's request
exp.rdf.stat.x2_qtree	Counter for the number of times a qtree cache object is refreshed on behalf of caller's request
exp.rdf.stat.z0_exists	
exp.rdf.stat.z12_itups	Counter for the number of times requested integrity tuples were not found
exp.rdf.stat.z13_etups	Counter for the number of times requested event permit tuples were not found
exp.rdf.stat.z14_ptups	Counter for the number of times requested permit tuples were not found
exp.rdf.stat.z15_rtups	Counter for the number of times rule tuples not found
exp.rdf.stat.z16_syn	Counter for the number of times RDF looked for a synonym, but none was found
exp.rdf.stat.z18_pptups	Counter for the number of times procedure parameter tuples were not found
exp.rdf.stat.z19_ktups	Counter for the number of times iikey tuples requested were not found
exp.rdf.stat.z1_core	
exp.scf.scd.acc	Expected average number of active cursors per session
exp.scf.scd.avgrows	Expected average number of rows returned

	in a select
exp.scf.scd.capabilities_num	Server capabilities–look up ima_server_capabilities in IMA_STATUS_LOOKUP
exp.scf.scd.capabilities_str	Server capabilities as a string
exp.scf.scd.csrflags	Cursor flags–look up ima_cursor_flags in IMA_STATUS_LOOKUP
exp.scf.scd.defcsrflag	
exp.scf.scd.fastcommit	0 if not using fast commit, non-zero if using fast commit
exp.scf.scd.flatflags	Server flattening options–look up ima_server_flatten_options in IMA_STATUS_LOOKUP
exp.scf.scd.gclisten_fail	Number of consecutive GCA_LISTEN failures; if too many, server closes
exp.scf.scd.irowcount	Number of rows to expect the first time through a select
exp.scf.scd.max_priority	
exp.scf.scd.min_priority	
exp.scf.scd.names_reg	0 if not registered with name server, non-zero if registered
exp.scf.scd.no_star_cluster	Zero if cluster optimizations should be done in Star; otherwise do not
exp.scf.scd.norm_priority	
exp.scf.scd.nostar_recovr	Zero if 2PC recovery should be done
exp.scf.scd.nousers	Maximum number of non-CL threads, including internal DBMS threads
exp.scf.scd.psf_mem	PSF memory pool size
exp.scf.scd.risk_inconsistency	User requested inconsistency risk (Recovery Server and DBMS may have different UDTs)
exp.scf.scd.rule_del_prefetch	
exp.scf.scd.rule_depth	Rule nesting depth

exp.scf.scd.rule_upd_prefetch	
exp.scf.scd.selcnt	Number of selects
exp.scf.scd.server.control.close	Control object: Any write causes the server to stop accepting new connections from non-privileged user.
exp.scf.scd.server.control.crash	Control object: Any write causes server to crash, immediately.
exp.scf.scd.server.control.open	Control object: Any write causes server to accept normal connections, and cancels any pending shutdown.
exp.scf.scd.server.control.shut	Control object: Any write causes server to stop accepting new non-privileged connections, and to shut down when the last user session exits.
exp.scf.scd.server.control.start_sampler	
exp.scf.scd.server.control.stop	Control object: Any write causes the server to shut down immediately no matter what the state of current connections.
exp.scf.scd.server.control.stop_sampler	
exp.scf.scd.server.current_connections	Number of current user sessions. Server is idle when count is zero.
exp.scf.scd.server.highwater_connections	Highwater count of number of connections for this server
exp.scf.scd.server.listen_mask	Server listen mask–look up ima_server_listen_mask in IMA_STATUS_LOOKUP
exp.scf.scd.server.listen_state	OPEN or CLOSED to non-privileged user connections
exp.scf.scd.server.max_connections	Maximum allowed number of user connections for this server
exp.scf.scd.server.pid	PID of this server as a decimal value, for VMS systems–converted to hex
exp.scf.scd.server.reserved_connections	Number beyond max_connections for privileged users to acquire
exp.scf.scd.server.shutdown_state	PENDING shutdown or OPEN
exp.scf.scd.server_name	Server name

exp.scf.scd.soleserver	Non-zero if this is the only server handling the databases in the dblist
exp.scf.scd.start.name	
exp.scf.scd.startup_time	
exp.scf.scd.state_num	Server activity–look up ima_server_activity in IMA_STATUS_LOOKUP
exp.scf.scd.state_str	Server activity as a string
exp.scf.scd.writebehind	Number of write behind threads
exp.scf.scs.scb_act_detail	More detailed information about the activity of the session (if known)
exp.scf.scs.scb_activity	The 'major' activity in which the session is engaged (if known)
exp.scf.scs.scb_appl_code	The application code of the session
exp.scf.scs.scb_client_connect	The 'connection string' requested by the client–including vnode, database, and server class information
exp.scf.scs.scb_client_host	Name of the VNODE (if known) on which this client session originated
exp.scf.scs.scb_client_info	The 'client information block'–including all ima_client_info columns and any user specified options
exp.scf.scs.scb_client_pid	PID (if known) of the original client process
exp.scf.scs.scb_client_tty	The terminal (if known) associated with the client process on the originating machine. This value will be the same as exp.scf.scs.scb_terminal for local connections
exp.scf.scs.scb_client_user	Username (if known) associated with the process that originated this connection
exp.scf.scs.scb_connect_limit	
exp.scf.scs.scb_crash_session	Control object: When written with a string of the decimal value of the session ID, causes the session to crash. This is dangerous, and may corrupt the server.
exp.scf.scs.scb_database	Database associated with the specified session
exp.scf.scs.scb_dblockmode	"Exclusive" or "shared"–the database locking mode for the session

exp.scf.scs.scb_dbowner	The owner of the database the session is using, empty if the session is in no database
exp.scf.scs.scb_description	
exp.scf.scs.scb_euser	The current effective user ID of the session
exp.scf.scs.scb_facility_index	The current facility for the session, as a number–look up ima_facility_index in IMA_STATUS_LOOKUP
exp.scf.scs.scb_facility_name	The current server facility as a string
exp.scf.scs.scb_gca_assoc_id	The GCA association ID for the session, plus 1 so assoc 0 is not used
exp.scf.scs.scb_group	Current effective group ID for the session
exp.scf.scs.scb_idle_limit	
exp.scf.scs.scb_index	The session ID for the session in a DBMS server–as a string representing a decimal value. This value is not represented as an integer because of overflow problems on 64-bit machines. To cross reference to the errlog.log, and so on, convert to hex
exp.scf.scs.scb_initial_connect_limit	
exp.scf.scs.scb_initial_idle_limit	
exp.scf.scs.scb_lastquery	
exp.scf.scs.scb_pid	The operating system PID of the server containing this session as a decimal value (for VMS systems, convert this value to hex)
exp.scf.scs.scb_priority	
exp.scf.scs.scb_priority_limit	
exp.scf.scs.scb_ptr	
exp.scf.scs.scb_query	The query being executed–this query consists of the query text passed down from the GCA message, and so will not have parameters 'in place'
exp.scf.scs.scb_remove_session	Control object: Writing with a string of the session ID causes the session to be terminated as if the client application had

	exited. This is the safe and preferred way to make a session go away.
exp.scf.scs.scb_role	Current effective role for the session
exp.scf.scs.scb_ruser	The operating system user that invoked the session
exp.scf.scs.scb_s_name	Our idea of the user who started the session
exp.scf.scs.scb_self	
exp.scf.scs.scb_terminal	Name of the terminal associated with this session in an operating system specific format. This value is valid for local connections only and will appear as 'batch' for Net connections–see ima_client info for the original user terminal.
exp.sxf.audit.audit_wakeup	
exp.sxf.audit.close_count	
exp.sxf.audit.create_count	
exp.sxf.audit.extend_count	
exp.sxf.audit.flush_count	
exp.sxf.audit.flush_empty	
exp.sxf.audit.flush_predone	
exp.sxf.audit.logswitch_count	
exp.sxf.audit.open_read	
exp.sxf.audit.open_write	
exp.sxf.audit.read_buffered	
exp.sxf.audit.read_direct	
exp.sxf.audit.write_buffered	

exp.sxf.audit.write_direct	
exp.sxf.audit.write_full	
exp.sxf.audit_writer_wakeup	
exp.sxf.db_build	
exp.sxf.db_purge	
exp.sxf.db_purge_all	
exp.sxf.mem_at_startup	
exp.sxf.queue_flush	
exp.sxf.queue_flush_all	
exp.sxf.queue_flush_empty	
exp.sxf.queue_flush_full	
exp.sxf.queue_write	

IMA_STATUS_LOOKUP

ima_server_state_mask	1	No thread switching
	2	Adding a new session
	4	Server will shutdown when last session exits
	8	Server WILL shutdown
	16	Server running with session accounting set
	32	Server is recording CPU statistic information
ima_server_internal_state_mask	0	Uninitialized
	16	Initializing
	32	Normal Processing mode
	37	Switching Sessions
	48	Server is Idle
	64	Server is closing down (set server shut)
	80	Server WILL shut down
	96	Server has encountered an error
ima_dmf_cache_status	1	Group
	2	Fwait

	4	Wbwait
	8	FCflush
	16	Shared Cache
	32	Pass abort in progress
	64	PrePass abort
ima_active_log_files	1	Primary Log file II_LOG_FILE
	2	Dual Log file II_DUAL_LOG
ima_log_header_status	1	Valid
	2	Recovery in progress
	3	EOF_OK
	4	BAD
	5	OK
ima_log_version	65537	Pre-OpenINGRES
	65538	OpenINGRES 1.1/03
ima_log_file_status	1	Logging system will attempt to use Dual IO
	2	Primary log has been opened
	4	Dual log has been opened
	8	Dual logging in progress
	16	Logging system will not use Dual IO

ima_logging_system_status	1	Online
	2	Consistency Point needed
	4	Logging system is full (stall condition)
	8	Force-Abort limit has been exceeded
	16	Logging system is in recovery mode
	32	Archiver is active
	64	Archiver is shutting down
	128	rcpconfig -imm_shutdown has been requested

	256	Archiver has shutdown, but is needed
	512	Purge in progress
	1024	Logging system is opening a new database
	2048	Logging system is closing a database
	4096	rcpconfig -shutdown has been requested
	8192	BCP has been written
	16384	Consistency point flush in progress
	32768	End Consistency point being processed
	65536	Consistency point has been completed
	131072	Consistency point has been signalled
	262144	Stall for start of CP
	524288	Online Checkpoint in progress
	1048576	Lartool used for manual abort
	2097152	Lartool used for manual commit

ima_lock_stall_type	1	Stall all except the Master CSP process
	2	Stall all user (logical) locks
ima_locking_system_status	1	AST_ENABLE
	2	CLUSTER_SYSTEM
	4	CSP_RUNDOWN
	8	IN_CSP_COMPLETION
ima_server_capabilities	1	Resource Control
	2	C2 security
	4	INGRES DBMS
	8	RMS
	16	INGRES Star
	32	RECOVERY server

ima_cursor_flags	1	DIRECT_UPDATE
	2	DEFERRED_UPDATE

ima_server_flatten_options	1	Optional flattening
	2	No flattening
	4	Flatten non aggregates
	8	Flatten non singleton
ima_server_listen_mask	1	Normal connections allowed
	2	Only special (priv) users connections permitted
	4	Server will close down when the last session exits

ima_server_activity	0	Uninitialized
	1	Initializing
	2	Shutting down
	3	Normal operation
	4	Participating in 2PC recovery

ima_async_mask	1	Timeout can happen
	2	Interrupts are allowed
	4	Thread stack has been corrupted
	64	Thread is marked for termination
	128	Interrupt is pending
	256	No transaction in progress
	512	Thread is a monitor session
	1024	Thread has fatal error
	2048	Request aborted
	4096	Thread started in mid-quantum
	8192	Event has completed
	16384	Interrupt received
	32768	Timed out
	65536	Collect CPU statistics for this session

ima_session_mode	1	Initiating

	2	Input
	3	Output
	4	Terminating
	5	Exchange
	10	Read

ima_thread_state	0	Unallocated block
	1	Computable
	2	Event wait
	3	MUTEX
	4	STACK_WAIT
	5	UWAIT
	6	CNDWAIT

ima_db_status	1	Database is active
	2	Database is journaled
	4	Control block is invalid
	8	Internal database - used for internal sessions
	16	Log records for this db are being purged to jnl
	32	OpenDB Pending
	64	CloseDB pending
	128	Recovery in progress
	256	Fast Commit
	512	Database invalid but being opened
	1024	CloseDB done
	2048	Tx waiting for OpenDB to complete
	4096	Db undergoing online ckp and is stalled
	8192	Db undergoing online ckp and is active
	16384	Db undergoing online ckp and is in final stage
	131072	Final Backup state - ckp only needs to update cnf
	262144	Error occurred on db during online ckp

ima_fcserver_cond	1	CPREADY
	2	CPWAIT
	3	RECOVER

ima_process_status	1	Process is still active
	2	Process is master recovery process (RCP)
	4	Process is archiver process (ACP)
	8	process is a fast commit server
	16	process is a server that has died abnormally
	32	process is a server
	64	process is checkpoint process
	128	process is dead and recovered
	256	process is a server with a shared buffer manager
	512	process is idle (connected to SBM but may exit)
	1024	process has died
	2048	process has been killed by logging system
	4096	FOREIGN_RUNDOWN
	8192	Process participates in Consistency Point

ima_tx_status	1	No log record written yet
	2	BT record has been written to log
	4	Tx is waiting for an online CKP
	8	Tx is protected from the ACP - BOF will not pass
	16	Records for this Tx should be archived
	32	No db changes will be made inside this Tx
	64	This Tx will never be the oldest for abort
	128	Tx needs recovery
	256	Tx has been chosed as the oldest for abort
	512	Tx is marked for abort by the RCP
	1024	Tx is marked for abort by the server
	2048	Tx is undergoing Pass Abort

	4096	Tx is accessing an inconsistent DB
	8192	Tx waiting for log event completion
	16384	Tx is part of a distributed Tx
	32768	Tx is prepared
	65536	REASSOC_WAIT
	131072	RESUME
	262144	MAN_ABORT
	524288	MAN_COMMIT
	1048576	LOGWRITER_THREAD
	2097152	SAVEABORT
	4194304	Tx needs space which it has not reserved
	8388608	Tx has lost the parent DBMS

ima_tx_wait_reason	1	FORCE
	2	FREE
	3	SPLIT
	4	HDRIO
	5	CKBDB
	6	OPENDB
	7	BCPSTALL
	8	LOGFULL
	9	FREEBUF
	10	LASTBUF
	11	BUFIO
	12	EVENT
	13	WRITEIO
	14	LOGSPACE

ima_lk_attribute	1	Physical
	2	Temporary
	4	Cancel
	8	Enqwait

	16	ConCancel

ima_lk_mode	0	Null
	1	Intent Shared
	2	Intent Exclusive
	3	Shared
	4	Shared-Intent-Exclusive
	5	Exclusive

ima_lk_state	1	GR
	2	CV
	3	WT

ima_lklist_status	1	Waiting
	2	Non Protected
	4	Orphaned
	8	EWAIT
	16	RECOVER
	32	MASTER
	64	ESET
	128	EDONE
	256	NOINTERRUPT
	512	SHARED
	1024	PARENT_SHARED
	2048	ENQWAITING
	4096	GDLK_SRCH
	8192	ST_ENQ
	16384	MASTER_CSP

ima_lk_type	1	DATABASE
	2	TABLE
	3	PAGE

	4	EXTEND_FILE
	5	BM_PAGE
	6	CREATE_TABLE
	7	OWNER_ID
	8	CONFIG
	9	DB_TEMP_ID
	10	SV_DATABASE
	11	SV_TABLE
	12	SS_EVENT
	13	TBL_CONTROL
	14	JOURNAL
	15	OPEN_DB
	16	CKP_DB
	17	CKP_CLUSTER
	18	BM_LOCK
	19	BM_DATABASE
	20	BM_TABLE
	21	CONTROL
	22	EVCONNECT
	23	AUDIT
	24	ROW

ima_facility_index	0	(none)
	1	(Compatibility Lib)
	2	(Abstract Data type Facility)
	3	(Data Manipulation Facility)
	4	(Optimizer Facility)
	5	(Parser Facility)
	6	(Query Execution Facility)
	7	(Query Storage Facility)
	8	(Relation Descriptor Facility)
	9	(System Control Facility)

	10	(Utility Facility)
	11	(Database Utility Facility)
	12	(General Communication facility)
	13	(Remote Query facility)
	14	(Transaction Processing facility)
	15	(Gateway facility)
	16	(Security Extensions facility)

ima_rdf_cache_status	1	Relation description cache is allocated
	2	LDB description cache is allocated
	4	Relation hash table is created
	8	Qtree hash table is created
	16	DEFAULT hash table is created

Appendix A: Environment Variables and Logicals

This section contains the following topics:

This appendix lists alphabetically the basic Ingres environment variables and logicals. Use this appendix as a reference for the chapter "Setting Environment Variables and Logicals."

Additional special purpose Ingres environment variables and logicals can be used in your installation, depending on the equipment and configuration.

DBNAME_ING

DBNAME_ING specifies a startup file of set commands to be executed whenever a user opens the specified database (*DBNAME*) *DBNAME*_ING can be defined installation-wide or locally.

Windows: DBNAME_ING affects any user who connects to the database specified by DBNAME using the QUEL Terminal Monitor.

UNIX: DBNAME_ING is effective in all user interfaces, including a terminal monitor.

VMS: The syntax for defining this logical is:

```
define DBNAME_ING path_to_file
```

DBNAME_SQL_INIT

DBNAME_SQL_INIT specifies a startup file of set commands to be executed whenever a user connects to the database specified by DBNAME using the SQL single-line Terminal Monitor. It can be set installation-wide or locally. For details on using DBNAME_SQL_INIT, see the chapter "Setting Environment Variables and Logicals."

DD_RSERVERS

DD_RSERVERS points to the directory hierarchy that is used by the Replicator servers.

Note: DD_RSERVERS is *not* a required variable. If it is not set, it defaults to the "rep" subdirectory under the Ingres installation directory.

Windows: By default, DD_RSERVERS is defined to be %II_SYSTEM%\ingres\rep. To define a different location, use, for example:

```
set DD_RSERVERS c:\rep2
```

UNIX: By default, DD_RSERVERS is defined to be the $II_SYSTEM/ingres/rep. To define a different location, use, for example:

C Shell:

```
setenv DD_SERVERS /usr/rep2
```

Bourne Shell:

```
DD_RSERVERS=/usr/rep2; export DD_RSERVERS
```

VMS: By default, DD_RSERVERS is defined to be the II_SYSTEM:[ingres.rep]. To define a different location, use, for example:

```
define DD_RSERVERS user_disk:[rep2]
```

The $DD_RSERVERS/servers (DD_SERVERS:[servers] for VMS) directory contains server1 through server10. These directories are used by the Replicator servers during runtime.

If you need your installation to support multiple Replicator configurations and/or multiple source databases, use DD_RSERVERS to access each Replicator environment. For each environment, create a DD_RSERVERS directory with the subdirectory servers. In the servers directory, create subdirectories server1 through server*N* where *N* is the number assigned to the Replicator Server.

II_4GL_DECIMAL

II_4GL_DECIMAL determines the decimal point character for 4GL source input. For example, in France, the decimal point character is a comma. The default value is ".". Valid values are "." and ",".

II_ABF_ASSUME_VALUE

In ABF applications, this variable forces parameters in the WHERE or ON clauses of a SELECT statement to be used as values (and thus have quotes where required). This is useful if the WHERE or ON clause includes a phrase that would otherwise be treated as a column name.

For details, see the Forms-based Applications Development Tools User Guide, the 4GL Statement Glossary chapter, Overview of the SELECT Statement, Where Clause.

II_AFD_TIMEOUT

II_AFD_TIMEOUT is used to set the number of seconds before timeout in Vision's Application Flow Diagram.

When you are using Vision's Application Flow Diagram, Vision periodically checks for changes made by concurrent developers that affect the diagram. If such a change has been made, Vision notifies you and gives you a chance to update the display.

You can set II_AFD_TIMEOUT to any value greater than 20, or 0 to disable the timeouts. Setting the value to much less than the default of 300 seconds causes Vision to time out and query the database often enough to visibly affect performance.

II_APPLICATION_LANGUAGE

II_APPLICATION_LANGUAGE is set to the language Application-By-Forms uses to translate string constants.

ABF allows you to internationalize your applications by translating the string constants you define for them. When you run the application, it normally uses the constants defined for your default language. You can override this by defining II_APPLICATION_LANGUAGE to the name of another language. For more information, see *Forms-based Application Development Tools User Guide*.

II_BIND_SVC_xx

II_BIND_SVC_xx is set to the bind address for Ingres tools connections to a named Ingres or Enterprise Access installation.

II_C_COMPILER (VMS)

II_C_COMPILER is set to the value of the C compiler in use for ABF. Valid entries are VAX11(VAX/VMS only), DECC, or NONE. II_C_COMPILER is defined at the system or group level. (Call technical support for information about how to change the C compiler used by ABF).

II_CHARSETxx

II_CHARSETxx is used to set the type of character set used for string operations. This is set during installation and cannot be changed without corrupting data.

II_CHECKPOINT

II_CHECKPOINT is an area set to the full file specification for the default checkpoint location. It is set during installation and cannot be changed, even during installation updates. Specific databases can designate alternate locations for checkpoints as a parameter to the createdb command.

II_CLIENT

II_CLIENT is set to "true" for client installations. Do not change II_CLIENT if it is set.

II_COLLATION

II_COLLATION determines the collation sequence for a non-Unicode-enabled database if one is not specified on the CREATEDB command (using the -l flag). II_COLLATION can be defined at the system level or redefined by the individual user at the process level.

Valid values are:

MULTI

Specifies the DEC Multinational character sequence

SPANISH

Specifies the Spanish alphabet's character sequence

collation_name

Specifies a custom collation sequence (non-Unicode)

More on collation sequences is described in Local Collation Sequences (see page 84).

II_CONFIG

II_CONFIG sets the full file specification of the Ingres files directory during the installation procedure. It cannot be changed.

II_CONNECT_RETRIES

II_CONNECT_RETRIES sets the number of times an application retries a connection before returning an error. The default is 1. An optional flag (d or D) causes a delay before the retry. For example: II_CONNECT_RETRIES=1d.

II_DATABASE

II_DATABASE is an area set to the full file specification for the default database location. It is set during installation and cannot be changed, even during installation updates. Specific databases can designate alternate locations as a parameter to the createdb command.

II_DATE_CENTURY_ BOUNDARY

II_DATE_CENTURY_BOUNDARY determines the inferred century for a date that is missing the century on input. It can be set to an integer value in the range of 0 < n < =100. If this environment variable is not set in the user's environment or if it is set to 0 or 100, the current century is used.

The century is determined by the following calculation:

```
if (current_year < II_DATE_CENTURY_BOUNDARY)
    if (input_year < II_DATE_CENTURY_BOUNDARY)
        output_year = input_year + current_century
    else
        output_year = input_year + previous_century
else
    if (input_year < II_DATE_CENTURY_BOUNDARY)
        output_year = input_year + next_century
    else
        output_year = input_year + current_century
```

For example, if II_DATE_CENTURY_BOUNDARY is 50 and the current year is 1999, an input date of 3/17/51 is treated as March 17, 1951, but a date of 03/17/49 is treated as March 17, 2049.

If the user enters the full four digits for the year into a four-digit year field in the application, the year is accepted as entered, regardless of the II_DATE_CENTURY_BOUNDARY setting.

II_DATE_FORMAT

II_DATE_FORMAT defines the format for date values. It is defined installation-wide.

If set, it replaces the default format (the US setting) with an alternative format. The following are valid II_DATE_FORMAT settings and their output formats:

Setting	Output Format
US (default)	dd-mm-yyyy
MULTINATIONAL	dd/mm/yy
MULTINATIONAL4	dd/mm/yyyy
ISO	yymmdd
ISO4	yyyymmdd
SWEDEN or FINLAND	yyyy-mm-dd

Setting	Output Format
GERMAN	dd.mm.yy
YMD	yyyy-mmm-dd
DMY	dd-mmm-yyyy
MDY	mmmm-dd-yyyy

For example, if the II_DATE_FORMAT setting is ISO, the dates are output as *yymmdd*. If you enter a date using a similar, six-character format, such as *mmddyy*, the date is interpreted on output as *yymmdd*. For example, if you enter the date March 9, 1912, as 030912 (*mmddyy*) and II_DATE_FORMAT is set to ISO, this date is interpreted as Sept. 12, 1903 (030912 as *yymmdd*). If the date you entered cannot be interpreted as a valid date in the output format, you receive an error message.

Note: When using the _date4() function (MULTINATIONAL4), the output format for the year value always returns '*yyyy*'.

For more information about valid date input and output formats, see the *SQL Reference Guide*.

II_DBMS_LOG

II_DBMS_LOG points to the location of the log file containing DBMS server-specific error messages. It is set installation-wide. The server reads it at startup time, so changing it after startup has no effect on an existing server. Most error messages written to this file are also logged to the installation-wide error log (errlog.log). Some server trace messages and statistics information are written to the II_DBMS_LOG file that are not written to errlog.log. The default, if not set, is $II_SYSTEM/ingres/files/errlog.log.

II_DBMS_SERVER

II_DBMS_SERVER points to the DBMS Server to which the client process automatically connects. It can be set installation-wide or locally. When II_DBMS_SERVER is set, application processes bypass the Name Server. Using II_DBMS_SERVER requires setting II_GC_REMOTE (see page 353) in the DBMS Server installation.

II_DECIMAL

II_DECIMAL specifies the character used to separate fractional and non-fractional parts of a number. Valid characters are the period (.) (as in 12.34) or the comma (,) (as in 12,34). The default is the period. This variable is set installation-wide.

Note: If II_DECIMAL is set to comma, be sure that when SQL syntax requires a comma (such as a list of table columns or SQL functions with several parameters), that the comma is followed by a space. For example:

```
select col1, ifnull(col2, 0), left(col4, 22) from t1:
```

II_DIRECT_IO (UNIX)

II_DIRECT_IO is specific to UNIX environment. If set, it causes disk reads and writes to be done directly to disk, circumventing the UNIX file buffer system. It is only available on some platforms.

II_DISABLE_SYSCHECK

II_DISABLE_SYSCHECK controls the execution of the **syscheck** command. The syscheck command checks the system to confirm the availability of the required resources for running Ingres. If II_DISABLE_SYSCHECK is set, syscheck is still called but will immediately exit without performing any checks and with a success return status. For more information on the syscheck command, see *Command Reference Guide*.

II_DML_DEF

II_DML_DEF defines the default query language for an installation as SQL or QUEL. It is used in installations that support both the SQL and QUEL database languages, in which case one such language can be designated as the default. Applications-By-Forms, Ingres 4GL, and Report-By-Forms are parts of the Ingres tools that use II_DML_DEF. SQL is the default if this is not set. It can be set installation-wide or locally.

II_DUMP

II_DUMP defines an "area" that specifies the default location of the dump log files used in online backup. (For more information on online backup, see the ckpdb description in the *Command Reference Guide*.) It is defined installation-wide during installation and cannot be changed.

II_EMBED_SET

II_EMBED_SET allows application programs to set a variety of features. It can be set installation-wide or locally. The acceptable values and the actions they specify are:

errorqry

Prints query that caused error.

savequery

Enables/disables saving of the text of the last query issued. Specify 1 to enable or 0 to disable. To obtain the text of the last query, you must issue the inquire_ingres(:*query*= querytext) statement. To determine whether saving is enabled, use the inquire_ingres(:*status*=savequery) statement.

printqry

Prints all query text and timing information to iiprtqry.log file in the current directory.

qryfile

Specifies an alternate text file to which query information is written. The default file name is iiprtqry.log.

printgca

Traces GCA messages to the iiprtgca.log file in the current directory.

gcafile

Specifies an alternate text file to which GCA information is written. The default file name is iiprtgca.log.

printtrace

Enables or disables saving of DBMS trace messages to a text file (specified by tracefile). Specify 1 to enable saving of trace output, or 0 to disable saving.

tracefile

Specifies an alternate text file to which tracepoint information is written. The default file name is iiprttrc.log.

sqlprint

Prints all errors and database procedure messages arising from embedded SQL statement.

eventdisplay

Print events as they return from DBMS.

dbmserror

Makes local (DBMS) error numbers the default error numbers returned to the application in errorno and the SQLCA's sqlcode.

genericerror

Makes generic error numbers the default error numbers returned to the application in errorno and the SQLCA's sqlcode.

prefetchrows

Specifies the number of rows to be "prefetched" when retrieving data using cursors. Valid arguments are:

0 (default)

Ingres determines the number of rows to prefetch.

1

Disables prefetching. Each row is fetched individually.

N (positive integer)

Specifies the number of rows to be prefetched.

II_ERSEND (UNIX)

II_ERSEND is set during the installation procedure to the full path name of the errlog.log file.

II_FORCE_HET

II_FORCE_HET must be set to TRUE to allow seamless communication between the local client and remote databases on 64-bit HP Tru64 Alpha (formerly DEC Alpha) systems. II_FORCE_HET must be set before the client program attempts communication. For the change to take effect, the Communication Server and Name Server must be stopped and restarted.

II_FRS_KEYFIND

II_FRS_KEYFIND allows users to turn off the keyfind frs function on table fields in read mode.

II_GC_REMOTE

II_GC_REMOTE allows or disallows a direct connection between the application process on the client machine and the Ingres DBMS process on the server machine, thus bypassing Name Server processing. The vnode attribute, connection_type, must be set to "direct" to enable direct connections. For additional information on this vnode attribute, see the *Connectivity Guide*.

II_GC_REMOTE needs to be set in the DBMS Server installation only.

Valid II_GC_REMOTE values are:

ON or ENABLED

On or Enabled allows direct connections that use the vnode attribute as described above, and rejects direct connections that are attempted using the II_DBMS_SERVER environment variable.

FULL or ALL

Full or All allows both vnode controlled and II_DBMS_SERVER controlled direct connections. You must avoid using this setting because it compromises network security. (No validation occurs to determine whether the client installation has permission to access the server installation.)

If II_GC_REMOTE is not set, or is not set to one of the above values, direct connections are disallowed.

II_GCA_LOG

II_GCA_LOG denotes the output destination for II_GC*x*_TRACE trace information for all Ingres components. Can be set to stdio or to the full path to a file.

II_GCx_TRACE

II_GCx_TRACE specifies environment variables and logicals for tracing:

II_GCA_TRACE

Sets the trace level for General Communications Architecture (GCA)

II_GCN_TRACE

Sets the trace level for Name Server (GCN)

II_GCC_TRACE

Sets the trace level for Communication Server (GCC)

II_GCD_TRACE

Sets the trace level for Data Access Server (GCD)

II_GC_TRACE

Sets the trace level for GCA Compatibility Library

II_GCCCL_TRACE

Sets the trace level for GCC Compatibility Library

Each variable can be set separately to a number from 0 to 4 specifying trace level, with levels 1 through 4 increasing in amount of information traced.

If any of these values are set to non-zero, trace statements for the appropriate items are logged in the II_GCA_LOG file.

II_GCD_LOG

II_GCD_LOG denotes the output destination for the Data Access Server (GCD) trace information. Setting this environment variable is preferable to using II_GCA_LOG because it is dedicated to the Data Access Server process.

II_GCN_LOG

II_GCN_LOG denotes the output destination for the Name Server (GCN) trace information. Setting this environment variable is preferable to using II_GCA_LOG because it is dedicated to the Name Server process.

II_GCNxx_PORT

II_GCNxx_PORT contains the connect address of a local installation's Name Server. This variable must not be reset.

xx = 2-character code defined by II_INSTALLATION

II_HALF_DUPLEX

II_HALF_DUPLEX, if set to 0 or undefined, enables Net duplex operation for SNA LU6.2 conversations. If set to 1, the Net half-duplex operation is enabled.

II_HELP_EDIT

II_HELP_EDIT, if set to any value, adds an extra operation, Edit, to menus encountered in help text screens in the forms systems. The Edit operation enables users to edit the help text in the screen with the text editor defined by ING_EDIT. II_HELP_EDIT is typically defined locally by the individual user.

II_INSTALLATION

II_INSTALLATION is a two-character code used to define a specific instance of Ingres. It is also referred to as an instance ID. This code is set during installation. To change it, rerun the installation program.

II_JOURNAL

II_JOURNAL is set to the full file specification for the default journal location. It is set during installation and cannot be changed, even during installation updates. Specific databases can designate alternate locations for journals as a parameter to the createdb command.

II_LANGUAGE

II_LANGUAGE determines what language is used for screen messages, menu items, and prompts. It is defined installation-wide. Valid values are as follows:

ENGLISH	English (Default)
DEU	German
ESN	Spanish
FRA	French
ITA	Italian
JPN	Japanese
PTB	Brazilian Portuguese
SCH	Simplified Chinese

II_LOG_DEVICE (VMS)

II_LOG_DEVICE specifies the name of the logging/locking pseudo device to which the logging/locking driver is connected. This value is QAA0: for production installations. For test installations, the value is QxA0:, where x is the first character of the group installation identifier. This logical is defined at the system or group level.

II_LP32_ENABLED

II_LP32_ENABLED enables a 32-bit client application to run against a 64-bit DBMS Server (reverse hybrid processing). Valid values are TRUE and FALSE.

The variable is valid only in Linux x86-64 (a64.lnx), Linux Itanium (i64.lnx), and HP Itanium (i64.hpu) environments.

Setting this variable before installing the Ingres 32-bit Support package files will cause runtime errors. Typically, the variable is set during installation by the iisusupp32 setup program when the associated 32-bit support files are installed for a reverse-hybrid installation.

After the 32-bit support files are installed, the II_LP32_ENABLED variable can be unset to re-enable 64-bit applications.

To set II_LP32_ENABLED, issue this command as the installation owner:

```
iisusupp32
```

To unset II_LP32_ENABLED, issue this command as the installation owner:

```
iisusupp32 -rmpkg
```

II_MONEY_FORMAT

II_MONEY_FORMAT defines the format of money output. It is set installation-wide. You can change the format by setting II_MONEY_FORMAT to a string with two symbols separated by a colon (:). The symbol to the left of the colon indicates the location of the currency symbol. It must be "L" for a leading currency symbol or a "T" for a trailing currency symbol. The symbol to the right of the colon is the currency symbol you want displayed. Currency symbols can contain up to 4 physical characters. For example:

Logical Definition	Result
L:$	$100
T:DM	100DM
T:F	100F

II_MONEY_PREC

II_MONEY_PREC specifies the number of decimal places to be displayed for money values. Valid values are 0, 1, and 2. The default is 2 (for decimal currency). It is set installation-wide. II_MONEY_PREC is applied when a money value is converted to a character string (for printing, for example). Extra decimal places are rounded. For example, if II_MONEY_PREC is set to 0, 9.50 is rounded to 10.

II_MSGDIR

II_MSGDIR sets the directory containing error and information messages. Users must not change this setting. The default file is the II_LANGUAGE setting, located at $II_SYSTEM/ingres/files/english.

II_NULL_STRING

II_NULL_STRING specifies the string used by a terminal monitor and other Ingres tools to represent the null value. The string consists of one to three characters. The default value is three blanks.

II_NUMERIC_LITERAL

II_NUMERIC_LITERAL, if set to FLOAT, tells Ingres to interpret the data type of floating-point constants or result variables to be float8 rather than decimal, which is the default. For more information, see Decimal Constant Semantics Change in the *Migration Guide*.

Note: II_NUMERIC_LITERAL should be used only temporarily for easing migration from Ingres 6.4.

II_NUM_OF_PROCESSORS

II_NUM_OF_PROCESSORS is set to the number of processors on the machine. The variable affects processing in the Ingres DBMS, particularly when waiting for mutually exclusive resources; a value larger than 1 causes a more compute-intensive strategy that may result in getting the resource faster; however, this strategy is not generally advisable on single processor machines.

II_NUM_SLAVES (UNIX)

II_NUM_SLAVES specifies the number of slave processes that a DBMS server creates to do disk operations. The default is two, but systems having applications with high I/O throughput requirements, many disks, or faster disk drives can increase the number of slave processes. The following example allows each new server that is started to have four slaves:

```
ingsetenv II_NUM_SLAVES 4
```

Stop and restart DBMS servers to reset the number of slaves. The maximum supported value is 30 and the minimum is 0. Setting this environment variable/logical to 0 degrades performance substantially.

Note: On UNIX implementations that use O/S threads, II_NUM_SLAVES is not set and there are no I/O slaves.

II_OO_TABLE_SIZE

II_OO_TABLE_SIZE allows you to increase the size of an internal COPYAPP table to support the copying of more application objects than is allowed by the default (4096 objects).

II_PATTERN_MATCH

II_PATTERN_MATCH determines the set of pattern matching characters for QBF and the ABF qualification function. These can be SQL-like, QUEL-like, or not set.

sql

The pattern matching characters used by QBF and the ABF qualification function are % _ [and]. The % is equivalent to the QUEL"*" and _ is equivalent to the QUEL "?".

quel

The pattern matching characters used by QBF and the ABF qualification function are * ? [and].

<not set>

If not set, or if it is set to anything other than sql or quel, the QUEL pattern matching characters are used.

II_PATTERN_MATCH can be set installation-wide or locally. The complete word, either sql or quel, must be entered. The setting for II_PATTERN_MATCH is case-sensitive.

II_PF_NODE

II_PF_NODE specifies the vnode name of a remote node to be managed through the Microsoft Windows Performance Monitor.

II_POST_4GLGEN

II_POST_4GLGEN specifies a command to be translated and executed by the Vision component following successful generation of Ingres 4GL code. The command is passed the full path name of the Ingres 4GL source file just generated. This logical is used only by sites that have installed the Vision product. For more information on this logical, see the *Forms-based Application Development Tools User Guide*.

II_PRINTSCREEN _FILE

II_PRINTSCREEN_FILE is used with the printscreen function. It specifies a default file name for the output file of the printscreen function. This environment variable/logical is usually defined locally by the individual user. If this environment variable/logical is set and no file name is sent to the printscreen function, no prompt is given and this file name is used. If not set, the user is prompted for a file name. If the file name "printer" is specified for II_PRINTSCREEN_FILE, the screen depiction is sent directly to the line printer. For more information, see the *Character-based Querying and Reporting Tools User Guide*.

II_PRINT_UTF8ALIGN

II_PRINT_UTF8ALIGN allows the default UTF-8 formatting used by the REPORT -f command to be turned OFF. When using the forms-based report utilities, this variable can be set to FILES to generate UTF-8 aligned files.

II_SQL_INIT

II_SQL_INIT (see page 66) can be set locally to the full path name of a file containing SQL commands. Typically, it is defined locally by the individual user. When a user with II_SQL_INIT set connects to a terminal monitor, the commands in the named file are processed.

II_STAR_LOG

II_STAR_LOG points to the location of the log file containing distributed DBMS server-specific error messages. It is set installation-wide. It is read by the server at startup time, so changing it after startup has no effect on an existing server. Most error messages written to this file are also logged to the installation-wide error log (errlog.log). Some server trace messages and statistics information are written to the II_STAR_LOG file that are not written to errlog.log. The default, if not set, is $II_SYSTEM/ingres/files/errlog.log.

II_SYSTEM

II_SYSTEM specifies an "area" that specifies the parent directory of the ingres directory, where many components of your installation are located. It must not be changed without reinstalling Ingres.

II_TEMPORARY

II_TEMPORARY specifies the directory where temporary files used by the Ingres tools are created. By default, these files are created in the user's current directory. II_TEMPORARY can be redefined installation-wide or locally.

II_TERMCAP_FILE

II_TERMCAP_FILE specifies an alternative termcap file to use. This file must be in Ingres termcap file format. It can be redefined installation-wide or locally.

II_TFDIR

II_TFDIR specifies the directory in which the Vision code generator searches for Ingres 4GL template files. This logical is used only by sites that have the Vision component installed.

II_THREAD_TYPE

II_THREAD_TYPE specifies whether Ingres uses internal threads or OS threads. Valid settings are:

OS (default)

Operating System threads

INTERNAL

Internal threads

This variable can be set during installation. For additional information on setting this variable for your platform, see the Readme file.

II_TIMEZONE_NAME

II_TIMEZONE_NAME specifies the world time zone that determines the installation's location for timing purposes. It is defined during installation.

II_TM_ON_ERROR

II_TM_ON_ERROR defines the action taken when an error occurs in a terminal monitor. The valid settings are continue (on error) or terminate (on error). If this environment variable/logical is not set, the default action for the (non-forms based) terminal monitor is to continue; the default for Interactive SQL or QUEL (the forms-based terminal monitor) is to terminate. It can be defined installation-wide or locally by the individual user.

II_TUXEDO_LOC

II_TUXEDO_LOC supports Ingres DTP for Tuxedo. Specifies the directory where the shared memory file is to be created. It must be the same for all servers in a group. If II_TUXEDO_LOC is not set, II_TEMPORARY is used.

II_TUX_SHARED

If II_TUX_SHARED is set to USER, the name of the shared memory segment used by Ingres is t<username>.tux; otherwise the name is t1.tux.

II_TUX_AS_MAX

II_TUX_AS_MAX is the maximum number of application and TMS servers that can be started. The default value is 32. A maximum of II_TUX_AS_MAX servers are permitted to attach to the Ingres shared memory segment.

II_TUX_XN_MAX

II_TUX_XN_MAX is the total number of transaction entries allocated in the shared memory segment. Each server that attaches the shared memory segment reserves II_XA_SESSION_CACHE_LIMIT transaction entries for its own use. The default value is 1024.

II_UNICODE_CONVERTER

II_UNICODE_CONVERTER sets the converter to be used for Unicode coercion. The DBMS Server carries out the coercion of Unicode data to local character set value based on the Unicode converter (or mapping table). Several converters are available in compiled and xml format, located in the directory $II_SYSTEM/ingres/files/ucharmaps. By default, Ingres automatically chooses the converter to use, based on the encoding used by the locale on your system.

You can manually set II_UNICODE_CONVERTER if Ingres is unable to set the converter correctly for your locale for coercing Unicode data to local character set value.

To manually set II_UNICODE_CONVERTER on UNIX based on locale

1. Type the following at the UNIX command prompt:

   ```
   locale
   ```

   The locale on your system is returned.

2. Use the locale value to find the character set on your system. For example, if locale is ja_JP.sjis, issue the following command:

   ```
   LC_CTYPE=ja_JP.sjis locale charmap
   ```

   The character set used by the system is returned.

3. Look up the character set in the file $II_SYSTEM/ingres/files/ucharmaps/charmapalias.xml. This file lists mapping IDs and their corresponding aliases. Find the mapping ID by finding the alias name value that matches the alias name. This mapping ID is the value of the converter.

4. Set the value of II_UNICODE_CONVERTER in the symbol table, as follows:

   ```
   ingsetenv II_UNICODE_CONVERTER mapping_id
   ```

Recycle the server by issuing the ingstop and the ingstart commands.

II_UUID_MAC (Windows)

II_UUID_MAC, when set to TRUE, uses an older algorithm for generating UUIDs on Windows.

For security reasons, the uuid_create function on Windows no longer uses a machine's MAC address to generate UUIDs. You can use the II_UUID_MAC environment variable if you do not need the level of security provided by the new uuid_create algorithm and for the sake of compatibility want to continue to use the MAC address of your machine when creating UUIDs.

II_VNODE_PATH

II_VNODE_PATH specifies a series of remote virtual nodes to be tried each time a user attempts to connect to the specified *vnode*. Node names are specified as strings to be appended to *vnode.* Commas must separate the strings.

For example, the following specifies that the remote virtual nodes mynode0, mynode1, mynode2, and mynode3 be tried each time a user attempts to connect to mynode:

II_MYNODE_PATH=0, 1, 2, 3

The names must be valid remote virtual nodes defined by the Ingres Network Utility or the Net Management Utility (netutil).

A single connection attempt begins with one of the nodes, chosen at random, and continues until all of the specified nodes have been tried. The number of connection attempts made before an error is returned to the user is determined by the value of II_CONNECT_RETRIES.

II_WORK

II_WORK specifies a directory that contains temporary files used by Ingres during the execution of certain queries such as sorts, copies, and modifies.

II_XA_TRACE_FILE

II_XA_TRACE_FILE specifies a file in which Ingres DTP (Distributed Transaction Processing) logs the events occurring through the TMXA interface, as well as any SQL performed against the Ingres DBMS. The user who starts the application servers must have write access to the file. For additional information about this file, see the *Distributed Transaction Processing User Guide*.

IIDLDIR

IIDLDIR specifies a directory that contains dynamically loaded libraries.

ING_ABFDIR

ING_ABFDIR defines the directory name for the top-level directory to contain Application-By-Forms object libraries and work files. Under this directory, a subdirectory is created for each database that contains ABF applications and, within that subdirectory, another subdirectory is created for each ABF application. The ING_ABFDIR directory level must have read, write, and execute permissions for group, owner, and world. If, however, the application designer redefines ING_ABFDIR to point to a directory of his own, he can optionally restrict permissions on the directory to read, write, execute to owner. It can be redefined installation-wide or locally.

ING_ABFOPT1

ING_ABFOPT1 specifies the name of a file, used by ABF, in which additional linker options can be placed. These options are not required to run ABF applications but can be required to include libraries, compiled objects, and other capabilities not available in ABF. Users can modify this file. For more information, see the *Forms-based Application Development Tools User Guide*. The individual user usually defines it at the local level.

ING_EDIT

ING_EDIT specifies the default editor invoked by various editor commands. The default for the entire installation is set during the installation procedure. Users can also set this in their local environment.

Windows: You must set ING_EDIT to the full path name of the editor. For example:

```
c:\tools\vi.exe
```

UNIX:

```
/usr/ucb/vi
```

VMS: If undefined, the callable EDT editor is called. ING_EDIT can be defined to be:

+EDT

    Calls the EDT editor.

+TPU

    Calls the TPU editor.

+LSE

    Calls the language sensitive editor.

The "+" symbol indicates that the callable version of the editor is to be used. This avoids creating a new subprocess in which to run the editor and results in much faster start up of the editor when called from a terminal monitor, ABF and other Ingres tools programs. Example:

```
DEFINE ING_EDIT "+TPU"
```

No qualifiers can be used. You must specify any non-default editor behavior in an editor startup file. For example, to use the Language Sensitive Editor in Fortran, define the following:

```
DEFINE ING_EDIT "+LSE" DEFINE LSE$COMMAND -  "SYS$LOGIN:LSE_STARTUP.LSE"
```

ING_PRINT

ING_PRINT specifies the default printer command issued by the Print function. The default is PRINT. It is usually defined locally by the individual user.

ING_SET

ING_SET specifies a quoted string or startup file of set commands to be executed whenever a user connects to the DBMS Server through an application, an Ingres tool, or a single-line terminal monitor. It can be set installation-wide or locally. For details on using ING_SET, see the chapter "Setting Environment Variables and Logicals."

ING_SET_DBNAME

ING_SET_DBNAME specifies a quoted string or startup file of set commands to be executed whenever a user connects to the database specified by *DBNAME.* Affects any user who connects to the specified database through an application, an Ingres tool, or a single-line terminal monitor. It can be set installation-wide or locally. For details on using ING_SET_*DBNAME*, see the chapter "Setting Environment Variables and Logicals."

ING_SHELL (UNIX)

ING_SHELL, If defined, contains the path name of the shell used by Ingres Menu and a terminal monitor when the shell operation is invoked.

ING_SYSTEM_SET

ING_SYSTEM_SET specifies a quoted string or startup file of set commands to be executed whenever a user connects to the DBMS Server through an application, an Ingres tool, or a terminal monitor. It is always global. For details on using ING_SYSTEM_SET, see the chapter "Setting Environment Variables and Logicals."

INGRES_KEYS

INGRES_KEYS specifies a mapping file to customize a user's keyboard layout for use with forms-based programs. For a complete description of the mapping file syntax, see *Character-based Querying and Reporting Tools User Guide*. This environment variable/logical is usually set in the user's local environment.

INIT_INGRES

INIT_INGRES can be set to the full file specification of a file containing QUEL commands. It is usually defined locally by the individual user. When a user with INIT_INGRES set connects to the QUEL Terminal Monitor, the commands in the named file are processed. For details on using INIT_INGRES, see the chapter "Setting Environment Variables and Logicals."

TERM (UNIX)

TERM is the terminal description for the terminal upon which you are executing UNIX programs. The TERM environment variable is not used directly, but it becomes the default for TERM_INGRES, if TERM_INGRES is not explicitly set. TERM is used by UNIX programs like vi, which can be called as subsystems from Ingres programs.

TERM_INGRES

TERM_INGRES contains the terminal designation for the terminal upon which you are executing one of the Ingres forms-based products, such as QBF or VIFRED. For a list of supported values, see *Character-based Querying and Reporting Tools User Guide*. It is usually defined locally by the individual user.

Windows: TERM_INGRES is most conveniently set in the symbol table to IBMPC.

UNIX: TERM_INGRES is most conveniently set in a .login or .profile file. TERM_INGRES takes precedence over TERM in defining the terminal type. It allows TERM to be defined differently for use by other UNIX programs such as vi.

VMS: TERM_INGRES is most conveniently set in a LOGIN.COM file.

Index