Ingres® 9.3

Command Reference Guide

INGRES

ING-93-CR-04

Contents

Chapter 1: Introducing Ingres Commands

This section contains the following topics:

Commands and Utilities

Many Ingres commands operate on the database as a whole. Some invoke Ingres querying and reporting tools, preprocessors, and utilities. Some utilities are special purpose programs or require special privileges to invoke.

Audience

This guide is a quick reference to Ingres® commands and system utilities.

The guide is for programmers and users of Ingres who have a basic understanding of how relational database systems work. In addition, the reader should have a basic understanding of the operating system.

Special Considerations

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

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 SQL-92. In addition, numerous vendor extensions are included. For details about the settings required to operate in compliance with ISO Entry SQL-92, see the *SQL 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*

UNIX Shells Used in this Guide

In this guide, command formats for the following UNIX shells are shown:

- C shell
- Bourne shell

For the Korn shell, use the Bourne shell syntax. Refer to your operating system shell documentation for any variations required on your particular system.

Command formats for the following UNIX operating system variants are shown where needed:

- BSD
- System V

Query Language Used in this Guide

The industry standard query language, SQL, is used as the standard query language throughout this guide. Ingres is compliant with ISO Entry SQL-92. For details about the settings required to operate in compliance with ISO Entry SQL-92, see the *SQL Reference Guide*.

Syntax Conventions Used in this Guide

This guide uses the following conventions to describe command and statement syntax:

Convention	Usage
`Monospace`	Indicates keywords, symbols, or punctuation that you must enter as shown.
Italics	Represent a variable name for which you must supply a 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.

Where to Issue Commands

You execute Ingres commands at the command line.

General Command Syntax

A command consists of one or more required command words, usually followed by one or more flags or parameters:

`command [flags] [parameters]`

A *flag* is a command option that consists of a letter preceded by a hyphen (-). A flag may stand alone (-f), or be followed by a parameter (-f*parameter*). Generally, there is no space between a flag and parameter.

Flags are shown in lowercase unless they are required to be uppercase. Uppercase flags (see page 17) may need special input syntax if the host operating system is case-insensitive.

A *parameter* is a command line option that is not a flag. A parameter can be the name of a database, a table or other object, or a value that specifies a particular use for a command.

In general, you can enter command options in any order. A few commands, however, require options in a specific order.

Standard Flags and Parameters

The following parameters and flags are common to many commands. Each command description in this guide indicates whether these parameters or flags are valid for that command.

The following syntax is typical for many commands:

```
command dbname|vnode::dbname[/server_class] [-fproduct]
[-uusername] [-Ggroupid] [-Rroleid] [other flags] [other parameters]
```

dbname

Identifies the name of a database. This parameter must precede all other non-flag parameters (with the exception of *vnode*::*dbname*).

***vnode*::**

Identifies the remote node on which the database is located. It must be followed by two colons (::) and the dbname parameter, with no intervening space.

The remote node can be specified as either of the following:

vnode_name

Is the virtual node name, as defined to Ingres Net, that points to the connection data and authorization data necessary to access a particular remote instance.

@host+

Is a "dynamic vnode" connection string that includes the connection data, user authorization, and attributes that are associated with a remote node. The format of @host+ is described in Dynamic Vnode Specification (see page 15).

server_class

Specifies the name of one of the Ingres servers or Enterprise Access products (for example, DB2 UDB). If you are accessing a distributed database or a non-Ingres database through an Enterprise Access product, you must specify the server_class. For valid values for *server_class*, see the *Connectivity Guide* or your Enterprise Access product documentation.

-fproduct

Specifies the name of a product parameter. In selected commands, the catalog modules for one or more products may be specified. The user interface catalogs are grouped into modules. Each Ingres tool requires a set of modules to operate. If you omit the product, the command reads the installation's authorization string and specifies all products that the authorization string permits.

The product parameter must be one of the following:

ingres

Processes catalogs for the Ingres tools (Applications-By-Forms, Query-By-Forms, Report-By-Forms, and Visual Forms Editor).

ingres/dbd

Processes catalogs for DBD.

vision

Processes catalogs for Vision.

windows_4gl

Processes catalogs for OpenROAD.

nofeclients

Directs the command not to process catalogs for any user interface products. You cannot use the nofeclients name in conjunction with the name of any valid user interface product; nofeclients is valid only in specified commands.

-u*username*

Specifies the effective user name for the session. Valid only for a privileged user, DBA, or sessions that have the db_admin database privilege. (Some commands, including ckpdb, rollforwarddb, verifydb, createdb, and destroydb, restrict the use of the -u flag to privileged users.)

Note: The -u flag does not assume the group of the effective user. Use the -G flag to distinguish between the real and effective user.

-G*groupid*

Specifies the group identifier for the session. After the system administrator defines a group identifier, a DBA can grant database permissions to the group. When you issue a command, specifying group ID (using the -G flag), the group's permissions are applied to the session.

To specify a group, you must be a member of the specified group identifier's user list, a system administrator, the DBA of the specified database, or a user that has the db_admin privilege.

If you omit this flag and there is a default group identifier specified for you, the default group identifier is assigned to the session. (Default group identifiers are assigned using accessdb.)

VMS: You must enclose this parameter in double quotation marks ("-G*groupid*").

-R*roleid*

Specifies the role identifier for an application image. After the system administrator defines a role identifier, a DBA can grant database permissions to the role ID. When you invoke an application and specify role ID (using the -R flag), the role permissions are applied to your session.

The *roleid* must be an existing role identifier. If the role identifier requires a password, you are prompted for the password. If you specify the -R flag, but omit both the role identifier and password, you are prompted for both. If no password is defined for the specified *roleid*, press the Enter key when prompted for the password.

Neither *roleid* nor password is validated if you are a system administrator, DBA for the specified database, or a user that has the db_admin privilege.

VMS: You must enclose this parameter in double quotation marks ("-R*roleid*").

For further information on groups and roles, see the *Database Administrator Guide*.

Dynamic Vnode Specification—Connect to Remote Node

When connecting to a remote node (using the *vnode*::*dbname* syntax), you can specify a dynamic vnode instead of a vnode name. The dynamic vnode specification includes the connection data, user authorization, and attributes that are associated with a remote node.

Note: A dynamic vnode can be used wherever a vnode is allowed, unless otherwise stated.

A dynamic vnode specification has the following format:

```
@host,protocol,port[;attribute=value{;attribute=value}][[user,password]]
```

@host

Identifies the network name or address of the node on which the remote database is located. The @ character is required because it identifies this specification as a dynamic vnode rather than a vnode name.

protocol

Identifies the network protocol to be used by the local node to connect to the remote node. For a list of protocols and their associated keywords, see the *Connectivity Guide*.

port

Identifies the listen address of the Ingres instance on the remote node.

attribute=value

(Optional) Is one or more additional connection, encryption, and authentication attributes for the connection. For a description of each attribute and its possible values, see the *Connectivity Guide*.

[*user*,*password*]

Identifies the user (login) name and password for the user on the remote system.

Note: The user and password are optional when creating a dynamic vnode, but must be enclosed in brackets if used. They are required if you want to authenticate using the default Ingres security mechanism.

Examples of dynamic vnode specification:

This command runs the terminal monitor (sql) and connects to node hosta using protocol tcp_ip to remote Ingres symbolic port II. The login and password are Johnny and secretpwd. The remote database name is customerdb:

```
sql @hosta,tcp_ip,II;[Johnny,secretpwd]::customerdb
```

This command establishes a direct connection by using the connection_type attribute:

```
sql @hosta,tcp_ip,II;connection_type=direct[Johnny,secretpwd]::customerdb
```

Uppercase Flags

Flags that must be entered in uppercase may need special input syntax when the host operating system is case-insensitive.

Windows: The Windows operating system passes uppercase flags with no special formatting needed. For example, to invoke Interactive Terminal Monitor with a group of sales, you could enter:

```
isql dbname -Gsales
```

UNIX: UNIX is case-sensitive and passes uppercase flags with no special formatting needed. For example, to invoke Ingres Menu with a group of sales, you could enter:

```
ingmenu dbname -Gsales
```

VMS: OpenVMS is case-insensitive and requires the addition of double-quotation marks around the uppercase flags. In OpenVMS, you must enclose all uppercase Ingres flags in double quotation marks. For example, to invoke Ingres Menu with a group of sales, use double quotes around the -G designation:

```
ingmenu dbname "-Gsales"
```

Schema Qualifier—Specify Ownership

A *schema* is a collection of database objects, such as tables. Each table, view, and synonym belongs to a schema that is determined when the object is created. The schema name corresponds to the user who owns the object. The schema name allows you to distinguish between objects with identical names but different owners.

You can specify a schema name for a table, view, or synonym on the command line to specify ownership. You use the following syntax:

```
schema.objectname
```

The period (.) must immediately follow the schema name and precede the object name, with no intervening spaces. Both the schema name and the object name can be delimited identifiers.

For example, to specify the table named "empinfo" having a schema name of dave, you would specify the table name as:

```
dave.empinfo
```

You do not use a schema name when referencing a table, view, or synonym; for example, you specify the table name as:

```
empinfo
```

The search looks first for an object with a schema corresponding to the current user; then it looks for an object owned by the DBA to which you have access. Lastly, if the object name begins with ii, the search looks for a system catalog with that name. For more information on schemas, see the *Database Administrator Guide*.

Delimited Identifiers on the Command Line

Delimited identifiers are database object names that are identical to reserved words, words that contain spaces, and non-alphanumeric characters that are disallowed in a regular identifier. If the installation allows mixed case names, you can also use delimited identifiers to distinguish among identical names with different case (for example, SALES and Sales).

On the command line, you use delimited identifiers if needed for names of tables, views, synonyms, schema, and authorization names (users, groups, and roles). For more information on allowable characters in delimited identifiers, see the *SQL Reference Guide*.

To create a delimited identifier, you must enclose the name in double quotation marks ("), dereference any embedded quotes, and use the appropriate number and type of delimiting quotes to pass it through your operating system. Use delimited identifiers on the operating system command line to specify database object names:

```
report my_database "Jane's table"
```

You must observe any operating system requirements for specifying quoted parameters, parameters containing embedded quotes, and parameters containing other characters that could be interpreted differently by the operating system. Depending on your operating system, you add delimiting and dereferencing quotes to a delimited identifier on the command line in order to pass it through the operating system with its own delimiting and embedded quotes (if any).

Examples: Delimited Identifiers

The following examples use the table names shown here:

Table Stored in Database	Delimited Identifier
Jane’s table	"Jane’s table"
"Expert" Table	"""Expert"" Table"

Windows: Surround delimited identifiers and their delimiting quotes with double quotes on the command line, and dereference the delimited identifier quotes, preceding them with a backslash (\):

```
report my_database "\"Jane's table\""
report my_database "\"\"\"Expert\"table\""
```

UNIX:

Bourne shell:
Surround delimited identifiers and their delimiting quotes with double quotes on the command line, and dereference the delimited identifier quotes, preceding them with a backslash (\):

```
report my_database "\"Jane's table\""
report my_database "\"\"\"Expert\"\" table\""
```

C shell:
Delimit all delimited identifier quotes and all other special shell characters, such as single quotes ('), spaces ( ), and colons (:), preceding them with a backslash (\):

```
report my_database \"Jane\'s\ table\"
report my_database \"\"\"Expert\"\" table\"
```

In some cases, strings contained inside delimited identifiers that contain special characters can be surrounded by double quotes instead:

```
report my_database \""Jane's table"\"
```

VMS: Surround delimited identifiers with a set of dereferenced double quotes on the command line. Also, you must dereference each embedded quote by doubling it (including any quotes required to dereference an embedded quote):

```
report my_database """Jane's table"""
report my_database """""""Expert""""" table"""
```

Delimited Identifiers Used on Authorization Parameters

You can use delimited identifiers to specify a *username* for the -u flag, a *groupid* parameter for the -G flag, or a *roleid* for the -R flag on the command line. A general example is:

```
sreport my_database myfile -u”user 5” -G”group 2”
```

Here are specific examples:

Windows:

```
sreport my_database myfile -u’”user 5”’ -G’”group 2”’
```

Windows NT:

```
sreport my_database myfile -u’”user5’””-G’”group 2’”
```

UNIX:

```
sreport my_database myfile -u’”user 5”’ -G’”group 2”’
```

VMS: In OpenVMS, you must also enclose the entire –G*groupid* parameter in double quotes:

```
sreport my_database myfile -u”””user 5”””“-G”””group2”””””
```

Delimited Identifiers and Case Sensitivity

By default, identifiers are forced to lowercase, and are therefore case-insensitive. The casing rules can be specified at installation time for delimited identifiers. The following settings are allowed:

- Ingres setting: lowercase (case-insensitive; forces all letters to lowercase).
- ISO Entry SQL-92 standard: mixed case (case-sensitive; preserves case for delimited identifiers); regular identifiers are uppercase (case-insensitive; forces all letters to uppercase).

If complying with ISO Entry SQL-92 standards, the system administrator should set delimited identifiers to mixed case.

Chapter 2: Using Ingres Commands

This section contains the following topics:

abf Command—Invoke Applications-By-Forms

The abf command invokes Applications-By-Forms (ABF), a forms-based interface for creating forms applications.

The abf command has the following format:

```
abf dbname |vnode::dbname[/server_class] applname [-w] [+wopen] [-5.0]
[-uusername] [-Ggroupid]
```

dbname

Specifies the name of the database, and the vnode and server_class, if required, as described in Standard Flags and Parameters (see page 13).

applname

Specifies the name of an ABF application. If omitted, ABF prompts for the name of the application.

-w

Causes the procedure names of an application to be checked for conflicts with system function names.

+wopen

Generates warnings if ABF detects statements that are not compatible with OpenSQL.

-5.0

Causes 4GL to be invoked in 5.0 compatibility mode.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13). Files that are created by ABF when using this flag are not owned by username, but by the user actually running the ABF process.

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

accessdb Command—Authorize User Access

The accessdb command invokes a forms-based interface by which the system administrator or another privileged user can authorize access to Ingres and individual databases. Accessdb is also used to extend databases to new locations.

The accessdb command has the following format:

```
accessdb [-uusername] [-vnode=vnode]
```

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-vnode=*vnode*

Specifies a vnode name as described in Standard Flags and Parameters (see page 13).

Note: A dynamic vnode is not valid on the -vnode parameter of the accessdb command.

aducompile Command—Install Customized Collation Sequence

Permission required: System administrator.

The aducompile utility compiles your description file into a binary file and installs that file as a collation sequence that can be used. Your new collation sequence will be in the following location:
$II_SYSTEM/ingres/files/collation/*collation_name*.

Note: On UNIX, everyone must have rights to read the new collation file.

The aducompile command has the following format:

```
aducompile description_filename language_filename [-u|-tu]
```

description_filename

Defines the name of the description file.

language_filename

Defines the name of the destination compiled file. The name must be unique to avoid overwriting existing collation files.

-u

Indicates the source is a Unicode collation table.

-tu

Indicates the source is an Ingres .uct map file. For more information on .uct map files, see the *System Administrator Guide*.

alterdb Command—Set Database Characteristics

Permission required: DBA or a privileged user running as the DBA.

The alterdb command sets journaling and other characteristics for a database. You can use alterdb to halt journaling for a specified database. To restart journaling, you must use the ckpdb +j command.

For further details about journaling options, see the *Database Administrator Guide*.

The alterdb command has the following format:

```
alterdb [-delete_invalid_ckp] [-keep=n] [-n|-i[ucollation_name]]
dbname[/server_class] [-target_jnl_blocks=n |-jnl_block_size=n |
 -next_jnl_file |-init_jnl_blocks=n |-disable_journaling |
 -delete_oldest_ckp] [-verbose] [-help]
```

-delete_invalid_ckp

Deletes all invalid checkpoints. Where there is a previous valid checkpoint, the journal and dump files associated with the invalid checkpoint are retained. For invalid checkpoints with no previous valid checkpoint, associated journal and dump files are removed.

-keep=*n*

Preserves the specified number of last valid checkpoints and deletes all older checkpoints (valid or invalid).

-n[*ucollation_name*]

Converts a non-Unicode database to a Unicode database with Normalization Form D (NFD). If no collation name is specified, the default collation (udefault) is used.

-i[*ucollation_name*]

Converts a non-Unicode database to a Unicode database with Normalization Form C (NFC). If no collation name is specified, the default collation (udefault) is used.

dbname

Specifies the database name, as described in Standard Flags and Parameters (see page 13). Specify one database name only. If required, identify the *server_class*.

-target_jnl_blocks=*n*

Specifies the number of journal blocks to be used for the database's journal file, where 32 <= n <= 65536. The current size can be obtained by the infodb Target journal size parameter.

If using Ingres Cluster Solution, this option has no effect on journal files created as part of a cluster merge.

-jnl_block_size=*n*

Specifies the size of each journal file block for the database, where *n* = 4096, 8192, 16384, 32768, or 65536 bytes. The current size can be obtained by the infodb Journal block size parameter.

Journaling must be off when issuing this command.

-next_jnl_file

Causes Ingres to start a new journal file for this database.

-init_jnl_blocks=*n*

Specifies the size of the first journal file created after a checkpoint is taken (with the ckpdb command), where 0 <= n <= current target journal size.

The target journal size is displayed by infodb and is the parameter set by the -target_jnl_blocks flag.

-disable_journaling

Halts journaling immediately, regardless of whether users are connected to the database.

Note: A side effect of using alterdb to disable journaling is that incorrect journaling status is displayed for tables. Tables display journaling as enabled—although journaling is disabled for the database—where you would expect enabled after next checkpoint.

-delete_oldest_ckp

Deletes the oldest available checkpoint, including related journals and dump files. The request fails if you attempt to delete the only remaining valid checkpoint.

-verbose

Displays system commentary to the standard output device as the alterdb operation continues. This parameter can be used with any one other alterdb parameter.

-help

Displays command syntax online.

arcclean Command—Purge Records from Replicator Shadow and Archive Tables

Permission required: This command must be executed from the DBA account or an account that has DB_ADMIN privilege on the databases.

The arcclean command purges unneeded records from the Replicator shadow and archive tables. It then modifies those tables back to their current storage structures. Use this command to reclaim disk space and improve performance.

After the Replicator Server processes data in the local queues, the records remain in the shadow and archive tables. This data continues to grow until you use the arcclean command to purge unneeded records. After arrclean is run, access to the shadow and archive tables is greatly improved.

To detect and resolve collisions, Replicator needs records of the last transactions on each row. Therefore, after arcclean is run, there will still be at least one shadow record for each record in the base table that has been touched by a replicated transaction. However, if there are any records in the input or distribution queues, the associated transactions in the shadow and archive tables also remain. To ensure that records remaining in input and distribution queues are not removed, the arcclean command selects records eligible for deletion and places them in a temporary table. If arcclean aborts, this temporary table should be removed automatically. If it is not removed, you can safely drop the table manually and then rerun arcclean.

You should run the arcclean command at intervals using the job scheduling mechanism of the operating system (such as cron on UNIX, and a batch job on OpenVMS). In UNIX, you can place the arcclean command in a cron file if $II_SYSTEM/ingres/bin is in the path. In OpenVMS, you can run the arcclean command from a batch job if you have executed the ingdbadef.com command file that defines the arcclean symbol.

Caution! Do not use arcclean if you do not have checkpoints and journals or other types of recovery mechanisms. The data purged from the shadow and archive tables can be used to aid recovery in a disaster.

The arcclean command has the following format:

Windows, VMS:

```
arcclean [vnode::]dbname "before_time"[--udba_name]
```

UNIX:

```
arcclean [vnode::]dbname 'before_time' [-udba_name]
```

[*vnode*::]*dbname*

Specifies the name of the database to be cleaned.

before_time

Indicates that all records in the shadow and archive tables dated before the specified date and time are to be purged. Provide the date and time in standard Ingres date and time format. On UNIX, place single quotes around the date and time; on VMS and Windows, use double quotes.

Caution! The date you specify should be before the last successful checkpoint or backup.

-u*dba_name*

Specifies the name of the database owner.

arclean Example

The following example purges from the database all records before a certain date.

Before issuing the arcclean command in this example, follow these steps to prepare the environment:

1. Make sure that you have valid checkpoints or backups of all databases to be cleaned. The date you specify in the arcclean command should be before the checkpoint or backup date.

   Removing records without a valid checkpoint could hinder recovery in the event of a system failure. For details, see the reconcil Command (see page 215).

2. Deny user access to all databases involved.

   There should be no new transactions during the cleaning process. Also, the shadow and archive tables need exclusive locks to the base tables during the modification procedure.Make sure the input and distribution queues are empty. You can do this by allowing the servers to run until they complete processing of all pending transactions.

   If records remain in the queues, the arcclean process will retain the relevant records in the shadow and archive tables on the local database. However, arcclean does not know which transactions are pending on other databases, and could therefore remove records on the local database that are required for an outstanding transaction on a target database. This situation may generate collisions when the outstanding transactions are distributed.

This command purges all records dated before 20-Feb-04 from the remote hq database:

Windows, VMS:

```
arcclean nyc::hq  -unyc_dba “20-feb-04”
```

UNIX:

```
arcclean nyc::hq  -unyc_dba ‘20-feb-04’
```

auditdb Command—Audit a Database

Permission required: DBA or system administrator. On VMS, to use this command against a database in a group level installation, you must have the VMS CMK RNL privilege.

The auditdb command prints selected portions of the journal for a database. It also creates an audit trail of the changes made to particular tables.

Auditdb does not necessarily give you a complete list of all transactions since the last checkpoint. Reasons for this are:

- Auditdb does not exclusively lock the database, so other users can complete a transaction while auditdb is running.
- In some cases, a completed transaction may not have been moved yet from the log files to the journal files.

If you need an accurate list of transactions since the last checkpoint, make sure all users exit the database before you run auditdb, or use the –wait flag. If you run auditdb with the –wait flag, and a large amount of unarchived information is in the log file, there will be a delay before the request is processed.

The auditdb command has the following format:

```
auditdb dbname[/server_class] [-a][-all] | [-table=tablename
{,tablename} [-file[=filename {,filename}]]] [-bdd-mmm-yyyy[:hh:mm:ss]]
[-edd-mmm-yyyy[:hh:mm:ss]] [#cn] [-iusername] [-inconsistent] [-wait]
[-uusername] [-help]
```

dbname

Specifies the database (one database name only), and the *server_class*, if required, as described in Standard Flags and Parameters (see page 13).

-a

Prints journal entries for the system catalogs.

-all

Prints everything in the journal file.

-table=*tablename* {,*tablename*}

Specifies a particular table or tables for which journal entries are to be printed. Up to 64 table names (and 64 file names if the –file flag is also used) can be specified on the command line. No spaces are allowed in the table list. If this flag is omitted, all tables in the database are audited.

This flag is not valid for system catalogs (-a flag).

The table name can be qualified with a valid schema name in the format *schema.tablename*, as described in Schema Qualifier (see page 18).

-file[=*filename* {,*filename*}]

Specifies that audit output is to go to one or more files. To use this option, you must specify the –table option.

If a file list is specified, the number of files must match the number of tables. The audit output of the first *tablename* goes to the first *filename*, and so on. No spaces are allowed in the file list.

If the –file flag is present without a list of file names, auditdb creates default file names of the form "*tablename*.trl" (the file extension is an abbreviation of "trail").

If a list of tables is specified without a list of files, output is presented to the standard output device.

This flag is not valid for system catalogs (-a flag).

The output files produced are in binary (bulk copy) format and contain rows appended to, deleted from, or copied into the tables specified. You can copy the output files into a table that has been created to have a row for each operation against the specified table. For more information, see the *Database Administrator Guide*.

-b*dd-mmm-yyyy*[:*hh*:*mm*:*ss*]

Prints journal entries for transactions committed after the specified date and time. If you specify a date and omit the time, the time defaults to 00:00:00 (midnight).

If you omit this parameter, auditdb lists transactions starting from the date and time of the most recent checkpoint.

-e*dd-mmm-yyyy*[:*hh*:*mm*:*ss*]

Prints journal entries for transactions committed before the specified date and time. If you specify a date and omit the time, the time defaults to 00:00:00 (midnight).

If you omit this parameter, auditdb lists transactions through the current system date and time.

#c*n*

Prints journal entries for transactions committed starting from an older checkpoint. The checkpoint number *n* must be a valid checkpoint number (as shown by the infodb command).

If you omit this parameter, auditdb lists transactions starting from the most recent checkpoint.

UNIX: In bash shell, you must place this option in quotes; otherwise characters after the # will be treated as a comment. For example:

```
auditdb empdata "#c1"
```

-i*username*

Prints journal entries for actions taken by the specified user.

-inconsistent

Lets you view journals that the database has marked as inconsistent. The audit will still fail if core catalogs are inconsistent.

-wait

Waits until journals are current before starting the audit. Auditing begins after all archiving is completed on the database, or after the archiver has copied all log file information up to the log file end-of-file when the auditdb request was initiated.

If a large amount of unarchived information is in the log file, there will be a delay before the request is processed.

Note: There is a small delay (typically a few seconds) between when a transaction is committed and when it is visible using auditdb.

-u*username*

Specifies the user for which journal entries are to be printed, as described in Standard Flags and Parameters (see page 13).

-help

Displays command options.

auditdb Examples

1. This command audits the empdata database:

   ```
   auditdb empdata
   ```

2. This command audits the empdata database, which contains the employee and address tables, and writes the output to the default files employee.trl and address.trl:

   ```
   auditdb empdata -table=employee,address -file
   ```

   This command does the same as the previous one, but specifies names for the output files:

   ```
   auditdb empdata -table=employee,address -file=aud2.trl,aud3.trl
   ```

3. This command creates an audit trail for the employee table in the empdata database, and then uses SQL commands to create a table and copy the default auditdb output files into it.

   ```
   auditdb empdata -table=employee -file sql empdata
   ```

   ```
   create table empaudit
     (date        date not null with default,
     usrname      char(32) not null with default,
     operation    char(8) not null with default,
     tranid1      integer not null with default,
     tranid2      integer not null with default,
     table_id1    integer not null with default,
     table_id2    integer not null with default,
     eno          I2,
     ename        char(10),
     age          I1,
     job          I2,
     salary       money,
     dept         I2);
   ```

   Windows:

   ```
   copy table empaudit () from 'C:\WINNT\Profiles\user1\employee.trl'
   ```

   UNIX:

   ```
   copy table empaudit () from "/usr/directory/employee.trl";
   ```

   VMS:

   ```
   copy table empaudit () from "dev:[directory]employee.trl";
   ```

blobstor Command—Copy a BLOB from a File to a Database

The blobstor command loads a binary large object (BLOB) into a column of an Ingres database, or stores the file name of the BLOB into a specified varchar (256) column. The BLOB is stored into a long byte column.

The blobstor command has the following format:

```
blobstor [-ttable -bblobcol -nnamecol] [-u -wwhereclause] [vnode::]dbname
imageFile
```

vnode

Specifies the virtual node where the database resides.

dbname

Specifies the database name.

imageFile

Specifies the file name of the binary object to be stored.

-t*table*

Specifies the name of the table to update.

-b*blobcol*

Specifies the name of the column to contain the BLOB. The default name is icedata.

-n*namecol*

Specifies the name of the column to contain the file name. The default name is icedata.

-u –w*whereclause*

Updates (-u) the BLOB in the location specified by the where clause (-w).

blobstor Examples

This command stores a picture logo.gif into a database style in the pic column of table images:

```
blobstor -t images -b pic -n fname style logo.gif
```

Note: To use blobstor, these two columns must exist in the table. For example:

```
create table images (fname varchar(256), pic long byte);\p\g
```

This command updates a picture into the database style in the picture column of table images:

```
blobstor -t images -b pic -n fname -u -w fname='logo.gif' style newlogo.gif
```

cacheutil Command—Show or Destroy Buffer Caches

Permission required: System administrator. On VMS, OpenVMS privileges.

The cacheutil utility returns information about shared memory buffer caches installed in the Ingres installation. The utility can show brief or detailed information about existing caches that are no longer in use. It can also destroy shared memory segments used for buffer caches that are no longer in use. Cacheutil does not destroy a buffer cache that is currently in use.

The cacheutil command has the following format:

```
cacheutil
```

cacheutil

Displays the cacheutil command prompt, where you can enter any of the following cacheutil commands:

list

Lists the existing shared buffer caches for the installation, including the size of the cache and the number of connected DBMS Servers.

show cache_name

Displays detailed statistics on the specified shared buffer cache.

destroy cache_name

Destroys the shared memory segment associated with the specified cache name.

This is needed on systems where shared segments are allocated in a manner such that they are not automatically released when all DBMS Servers connected to them are brought down. On these systems, if a DBMS with a shared memory cache fails or is brought down in an unsupported manner, the shared memory segment cannot be automatically cleaned up by the recovery system.

In systems where shared segments are automatically released, when the failed server is restarted, it will automatically clean up the old shared segment. In this case cacheutil is not necessary to release the shared memory.

However, if no server will be restarted that specifies the same cache name as the orphaned cache, then the shared segment must be cleaned up through the destroy option of cacheutil.

help

Displays help on the cacheutil utility.

exit

Closes the cacheutil utility.

catalogdb Command—List Databases That You Own

Permission required: System administrator, DBA.

The catalogdb utility is a forms-based interface that enables you to list your databases, the databases that you can access, the location names known to the system, the extensions made to your databases, and your user capabilities. For information on how to modify these attributes, see accessdb Command (see page 27).

The utility requires you to define the type of terminal you are using.

The catalogdb command has the following format:

```
catalogdb [-uusername] [-vnode=vnode]
```

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-vnode=*vnode*

Specifies a vnode name as described in Standard Flags and Parameters (see page 13).

Note: A dynamic vnode is not valid on the -vnode parameter of the catalogdb command.

catalogdb Examples

This command lets you browse through data on your own account and databases:

```
catalogdb
```

This command lets you, as system administrator, browse the data for another user:

```
catalogdb -uPeter
```

cbf Command—Start Configuration-By-Forms

Permission required: Access to the directory where the utility is located.

The cbf command starts the Configuration-By-Forms (CBF) utility. The CBF utility displays current values of the server parameters and lets you change them.

With CBF you can configure various components of the installation, select which databases can be accessed by a DBMS Server, reformat transaction log files and enable or disable dual logging, reconfigure protocol accesses for the Communications Server, set a new value of any configuration parameter, or restore the factory default, automatically calculate configuration parameters derived from other parameters, protect any derived parameter from further change, and run a system check for sufficient resources on a new configuration.

The cbf command has the following format:

```
cbf [-host=name] [-node=nodename]
```

-host=*name*

Specifies the name of the remote NFS client installation to be configured. When using this parameter, system resource checking must be disabled.

-node=*nodename*

Specifies the node to be configured. This parameter is valid in a cluster installation only.

ckpdb Command—Checkpoint a Database

Permission required: System administrator, DBA, or an Ingres user with the operator privilege. On VMS, to checkpoint a database in a group level installation, you must have the VMS CMKRNL privilege.

The ckpdb command checkpoints a database or selected tables in a database.

The command creates a new checkpoint for the specified database. If a table list is specified, only the tables on the table list are included in the checkpoint. If journaling is enabled for the database, all journal entries up to this checkpoint are marked as expired. Checkpointing takes place online (while the database is in use) and is transparent to users.

The ckpdb command creates the checkpoint, and then copies (to the dump file) the log records of any changes to the database that occurred during the checkpoint procedure. Rollforwarddb uses the dump file when it recovers a database that was checkpointed online.

Ingres knows whether the checkpoint is for a table or a database, and prevents attempts to roll forward an entire database from a table checkpoint. For table checkpoints, an infodb display of the mode field of the Journal Checkpoint History and Dump Checkpoint History will indicate TABLE.

By default, the ckpdb command sequentially checkpoints data locations one at a time. A database with more than one data location can be checkpointed in parallel.

The ckpdb command has the following format:

```
ckpdb dbname[/server_class] [-d] [+j|-j] [-l] [#m[n]] [-mdevice {, device}]
[-table=tablename {, tablename}] [-v] [+w|-w] [-timeout=mm:ss] [-uusername]
[-help]
```

dbname

Specifies the database (one database name only) to be checkpointed, and the *server_class*, if required, as described in Standard Flags and Parameters (see page 13).

-d

Destroys all previous checkpoint and journal files.

+j|-j

Enables or disables journaling for a database. When this flag is not specified, current journaling status of the database is maintained. If you specify this flag, the checkpoint is performed offline.

-l

Takes an exclusive lock on the database. If you specify this flag, the checkpoint is performed offline (while the database is not in use), which requires the database to be locked.

In an interactive session, if you specify the -l flag to perform the checkpoint offline, then you can also specify the +w or -w flag.

#m[*n*]

Checkpoints *n* locations at a time to disk, for a multi-location database.

-m*device* {, *device*}

Writes the checkpoint to the specified tape device. If a list of tape devices is supplied, parallel checkpointing is used for a multi-location database.

You can write one checkpoint only per tape. It is not necessary to mount the tape device. (When you restore a checkpoint that was created using the ckpdb -m command, you must use the rollforwarddb +c command.)

The –m option is not valid on Windows.

-table=*tablename* {, *tablename*}

Specifies a list of tables to be checkpointed. When specifying multiple tables, do not use a space between table names. Table checkpoint is not allowed for system catalogs.

To use this parameter, the database must be journaled. Do not use the +j or –j flag with -table.

-v

Indicates verbose mode, which displays interim messages as checkpointing proceeds.

+w|-w

Waits or does not wait for the database to be free (not in use) before performing the checkpoint. Use this flag only if you have specified the +j, -j, or -l flag. The default is -w.

This flag cannot be used if the checkpoint is performed online. An offline checkpoint requires the database to be locked.

If you specify +w, ckpdb waits as long as necessary for the database to become free for locking and checkpointing. If you specify -w, and the database is busy, an error is returned.

-timeout=*mm*:*ss*

Waits the specified number of minutes for active sessions to complete. If the active sessions do not complete in the specified time, the checkpoint is abandoned.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-help

Displays command syntax online.

ckpdb Examples

This command checkpoints and initiates journaling on the empdata database:

```
ckpdb +j empdata;
```

This command checkpoints the tables employee and dept:

```
ckpdb empdata -table=employee,dept
```

This command checkpoints the empdata database and retains only the newest checkpoint:

```
ckpdb empdata -d;
```

This command checkpoints the empdata database to tape:

UNIX:

```
ckpdb -m/dev/rmt0 empdata
```

VMS:

```
ckpdb -mMTA0: empdata
```

compform Command—Compile a Form

The compform command compiles a form that is stored in a database and places it in a text file.

When you are developing applications, you often must compile a form for use with embedded SQL or other embedded query languages. Compiling a form reduces start-up time when the form is then used in an embedded query language program.

The compform command performs the same operation as the Compile operation on the VIFRED Utilities menu.

To describe the form, the compform command produces a C language structure, by default.

The compform command has the following format:

```
compform dbname |vnode::dbname[/server_class] form filename [-m]
[-uusername] [-Ggroupid]
```

dbname

Specifies the name of the database. Also specifies the *vnode* and *server_class*, if required, as described in Standard Flags and Parameters (see page 13).

form

Specifies the name of the form. You can compile only one form at a time.

filename

Specifies the name of the text file into which the compiled form is placed.

-m

VMS: Compiles a form into OpenVMS macro code. This flag is set by default on OpenVMS machines.

Unless you specify this flag, compform compiles the form into C language code.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

compform Example

This command compiles the employees form, which is stored in the emp database, into a C language data structure and places it in the file empform.c:

```
compform emp employees empform.c
```

Before you can link the compiled form to your application, you must translate the compiled form into object code. For more information, see the *Character-based Querying and Reporting Tools User Guide*.

convrep Command—Upgrade the Replicator Data Dictionary

The convrep command converts the Ingres Replicator data dictionary for use with the current release of Ingres.

The convrep command has the following format:

```
convrep [vnode::] dbname [-udba_name]
```

vnode

Specifies the virtual node where the database resides.

dbname

Specifies the database name.

-udba_name

Specifies the name of the database owner.

convtouni Command—Convert Character Data to Unicode

Permission required: System administrator.

The convtouni utility converts character data in a database to Unicode, which lets you conveniently transform data in local encoding to Unicode. The utility converts all columns of data type char to nchar, and varchar to nvarchar.

Note: Convtouni should be run on a database that has Unicode support enabled. To enable Unicode support on a non-Unicode database, run alterdb with the –n option. For details, see alterdb Command (see page 29).

This utility creates the following files:

- An SQL script (ctouout.sql) that contains appropriate alter table statements to be executed
- An execution script (ctouexec.ing on UNIX and ctouexec.bat on Windows) for executing the SQL commands on the database

After running the convtouni utility, examine the SQL script and then execute the execution script.

When you run convtouni in automodify mode, the execution script is created in a temporary directory and executed immediately.

By default, all columns of all tables that have char or varchar columns are altered and modified. If you specify a table list, only the listed tables are altered (if they have char or varchar columns).

The convtouni command has the following format:

```
convtouni dbname [-param_file=filename] | [[-uuser] [-P] [-Ggroupid] [-dest=dir]
[-sqlfile=filename] [-automodify] [-col=column [-col=column]...] [{tables ...}]]
```

dbname

Specifies the name of the database to be exported.

-param_file=*filename*

Specifies the command file that contains the options to the convtouni command. If used, do not specify other options on the convtouni command. The file must contain all options and each parameter (dbname, table, user, and so on) must be on a separate line.

-u*user*

Specifies the effective user for the session.

-P

Specifies the password if the session requires one.

-G*groupid*

Specifies the group ID of the user.

-dest=*dir*

Specifies the directory the output SQL file will be written to.

-sqlfile=*filename*

Specifies the name of the output SQL file.

-automodify

Modifies the tables immediately by creating the execution script in a temporary directory and executing it.

-col=*column*

Specifies the column to convert, in *tablename.columnname* format. Each column must be preceded by –col=. Only the specified columns are changed; the rest of the char and varchar columns remain unchanged.

Note: No more than 100 objects can be specified. This limit can be raised by modifying the utexe.def file. For more information, see the *Database Administrator Guide*.

tables

Specifies a list of tables to convert. If no list is specified, but the -col= option is, then all specified columns of all tables owned by the user are altered. However, for the tables specified on the –col= option, only the specified columns are converted to Unicode.

Note: No more than 100 objects can be specified. This limit can be raised by modifying the utexe.def file. For more information, see the *Database Administrator Guide*.

convtouni Examples

Given a database userdb with tables tab1, tab2, tab3, and tab4 with char and varchar columns, the following command creates statements to convert all columns of all tables to Unicode.

```
convtouni userdb
```

The following command converts three columns in the tab1 table and all columns of tab2 table of the userdb database into Unicode. The sqlfile=myfile.sql and the executable script is generated in directory /users/ingres/myloc:

```
convtouni userdb -dest=/users/ingres/myloc sqlfile=myfile.sql -col=tab1.col1
-col=tab1.col2 -col=tab1.col3 tab2
```

The following command immediately executes the above commands:

```
convtouni userdb -automodify -col=tab1.col1 -col=tab1.col2 -col=tab1.col3 tab2
```

The following command converts all columns of all tables, but only col1 and col2 of the tab1 table:

```
convtouni userdb -automodify -col=tab1.col1 -col=tab1.col2
```

copyapp Command—Copy an Application to Another Database

The copyapp commands copy an application, created with Applications-By-Forms or Vision, from one database to another.

The **copyapp out** command copies information about the application and its objects to an intermediate text file. The **copyapp in** command transfers the information from the text file into a database. For more information on these commands, see the *Forms-based Application Development Tools User Guide*.

The copyapp out command has the following format:

```
copyapp out dbname |vnode::dbname[/server_class] applname [-ddirname]
[-tintfilename] [-lfilename] [-uusername]
```

dbname

Specifies the name of the database. Also specifies the *vnode* and *server_class*, if required, as described in Standard Flags and Parameters (see page 13).

applname

Specifies the name of the application to be copied.

-d*dirname*

Specifies the directory in which to create the intermediate text file; the default is the current directory.

-t*intfilename*

Specifies the intermediate text file filename; the default file name is iicopyapp.tmp.

-l*filename*

Creates a file containing the names of the source files to be copied. For Vision applications, the list includes only custom frames.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

The copyapp in command has the following format:

```
copyapp in newdbname |vnode::dbname[/server_class] [-nnewapplname] [-ddirname]
intfilename [-lfilename] [-c] [-p] [-q] [-r] [-s[dirname]|-a[dirname]]
[-uusername]
```

newdbname

Specifies the name of the database into which the application is to be copied. Also specifies the *vnode* and *server_class*, if required, as described in Standard Flags and Parameters (see page 13).

-n*newapplname*

Specifies the name to be assigned to the application in the new database. The default is the same name as in the old database.

-d*dirname*

Specifies the directory where the intermediate text file is located; the default is the current directory.

intfilename

Specifies the name of the intermediate file previously created by the copyapp out. This will be iicopyapp.tmp unless a different intermediate file name was designated with the –t flag of copyapp out.

-l*filename*

Creates a file containing a list of source files that were copied (or processed if the –a flag was specified). For Vision applications, the list includes only custom frames.

-c

Specifies that the intermediate text file should be deleted.

-p

Suppresses messages about name conflicts. The default is to display messages.

-q

Specifies that copyapp will be performed as a single transaction. If there is a duplicate name conflict, all changes are rolled back. If you specify the –q flag, copyapp locks system catalogs; for this reason, you should not specify –q when users are connected to the database. In addition, the transaction is logged in the log file; you should be sure that the log file is large enough to accommodate the copyapp transaction.

-r

Specifies that objects with the same name should be replaced (overwritten). By default, duplicate names are not overwritten; instead, the copy is not completed and terminates with an error message.

-s[*dirname*]

Specifies a new directory for source files. If dirname is omitted, the current working directory is used as the new application's source directory. This flag transfers 4GL source for custom Vision frames, but does not transfer source for non-custom Vision frames. Instead, it marks these frames as new, and source for these frames is regenerated on the next Go or Image operation.

This flag is intended for Vision applications. You cannot specify both the –a and the –s flags. For Vision applications that contain non-custom frames, be sure to use the -a or –s flag; if you do not, all frames are marked as custom frames.

-a[*dirname*]

Specifies a new source directory for the application, but does not copy source files. If dirname is omitted, then the current working directory is used as the new application's source directory. Any Vision frames are marked as new; source for these frames is regenerated on the next Go or Image operation.

This flag is intended for Vision applications. You cannot specify both the –a and the –s flags.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters **(see page** 13**).**

copyapp Example

The following commands copy the Vision new_emp application from the employee database to the employee2 database, use the default intermediate text file, and use the current working directory as the source directory for the new application:

```
copyapp out employee new_emp

copyapp in -a employee2 iicopyapp.tmp
```

copydb Command—Copy and Restore a Database

The copydb command creates command files containing the SQL statements required to copy and restore a database. The command creates the following two command files in the current directory:

- copy.out contains SQL commands to copy all tables, views, and procedures owned by the user into files in the specified directory.
- copy.in contains SQL commands to copy the files into tables, recreate views, procedures, and indexes, and perform modifications.

To copy the database, you must execute the SQL commands in the copy.in and copy.out command files.

The name of a file created by copy.out consists of the name of the table followed by an extension made up of the first three letters of the owner's login name. If file names collide, a unique digit replaces the last character of the table name segment.

Note: It is important that the database be recreated with copy.in before doing any work (for example, creating tables, forms, applications, or reports) in the new database. After recreating a database, be sure to run sysmod to optimize storage structures.

System catalogs cannot be copied using copydb. Use unloaddb to copy a complete database, including system catalogs.

Copydb can be used to change ownership of tables. For details, see the *Database Administrator Guide*.

Note: When copydb is run from an Ingres 2006 Release 2 or later installation against an older version of Ingres, the copy.in script generated will contain the data type INGRESDATE or ANSIDATE instead of DATE for any date columns in create table statements.

The copydb command has the following format:

```
copydb [-param_file=filename] | [dbname|vnode::dbname[/server_class] [-c]
[-uusername] [-Ggroupid] [-group_tab_idx] [-parallel]] [-journal]
[-P] [-source=dirname] [-dest=dirname] [-ddirname] [-with_tables] [-with_modify]
[-nodependency_check] [-with_data] [-all] [-order_ccm] [-with_index]
[-with_constr] [-with_views] [-with_synonyms] [-with_events] [-with_proc]
[-with_reg] [-with_rules] [-with_alarms] [-with_comments] [-with_roles]
[-with_sequences] [-no_seq] [-with_permits] [-add_drop] [-infile=filename]
[-outfile=filename] [-relpath] [-no_loc] [-no_perm] [-noint] [-no_persist]
[-no_repmod] [-no_rep][-online] {tablename|viewname}] [-help]
```

-param_file=*filename*

Reads *filename* for all other command line flags, database names, and any other command line arguments. This file must contain only one flag per line (see the examples that follow). If this flag is specified, no other flags or arguments can appear on the command line; they must, however, appear in the specified file.

dbname

Specifies the name of the database and, if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-c

Creates a printable data file. This is useful for transporting databases between computer systems whose internal representations of non-ASCII data differ. (When you restore a database from a file created using the –c flag, the copy command automatically converts data stored in this format back to the appropriate type.)

Copydb cannot represent the following types of data using printable characters: (1) binary data stored in varchar columns, and (2) user-maintained logical keys.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13) and Schema Qualifier (see page 18).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13). You must enclose this parameter in double quotation marks ("*-Ggroupid*").

-group_tab_idx

Builds indexes in the command file immediately after the respective table creation. Without this flag, all indexes are created for all tables toward the end of the command file. The usermod command uses this flag to limit the loss of non-persistent indexes if it encounters a failure.

-parallel

Creates indexes using the parallel index creation syntax (to build multiple indexes concurrently).

-journal

Replaces the SET NOJOURNALING statement in the copy.in scripts with the SET JOURNALING statement, and disables specifying the WITH NOJOURNALING option on each CREATE TABLE statement in copy.in script. When using the –journal flag, fastload is not possible when loading the tables.

-P

Prompts for password if the session requires one.

-source=*dirname*

Specifies the directory that contains the data files and from which copy.in will be run. An empty *dirname* specification (“”) denotes the current directory. The –source specification overrides a –d specification for the copy in file.

If a source is specified without a destination (no –d or –dest), the default copy out directory is used.

The source directory specification is not checked for validity or existence. This allows the scripts to be moved to another machine.

-dest=*dirname*

Specifies the directory where the data files created by copy.out will be stored. An empty *dirname* specification (“.”) denotes the current directory. The –dest specification overrides a –d specification for the copy out file.

If a destination is specified without a source (no –source) then the default copy in directory is used.

The destination directory specification is not checked for validity or existence. This allows the scripts to be moved to another machine. The destination directory **must** be different from the database directory, $II_DATABASE/ingres/data/default/dbname, because the files have the same names as the table files.

-d*dirname*

Stores the copy.in and copy.out files in the specified directory instead of the default current directory. The file name must be fully specified.

-with_tables

Prints only the CREATE statements.

-with_modify

Prints only the MODIFY statements.

-nodependency_check

Adds the WITH NODEPENDENCY_CHECK option to any MODIFY commands generated. This option forces a table modify operation and destroys indexes needed for constraints.

Important! If you use this option, you must preserve or recreate the table structure necessary to enforce the constraints.

-with_data

Prints only the COPY statements.

-all

Prints all the statements related to the database.

-order_ccm

Determines the order in which the COPY and MODIFY statements are written for the table. The default is to modify and then copy. If –CCM is specified, the order is to copy and then modify.

-with_index

Prints statements only related to index.

-with_constr

Prints statements only related to constraints, such as ALTER TABLE statements.

-with_views

Prints statements only related to views.

-with_synonyms

Prints statements only related to synonyms.

-with_ events

Prints statements only related to events.

-with_proc

Prints statements only related to procedures.

-with_reg

Print statements only related to registration.

-with_rules

Prints statements only related to rules.

-with_alarms

Prints statements only related to security alarms.

-with_comments

Prints statements only related to comments.

-with_roles

Prints statements only related to roles.

-with_sequences

Prints statements only related to sequences.

-no_seq

Does not print sequence related statements.

-with_permits

Prints statements only related to permits.

-add_drop

Writes a DROP statement also, before writing the CREATE statements. This is useful when the scripts are run repeatedly in case of errors, and tables are already created.

-infile=*filename*

Specifies an input file name for the copy.in file, so user can run copydb with different options and give different names for infile.

-outfile=*filename*

Specifies an output file name for the copy.out file.

-relpath

Removes the paths from the file names; the files will thus be created and copied from the current directory.

-noint

Runs copydb uninterrupted for all the tables.

-no_loc

Does not write the LOCATION clause for CREATE TABLE, CREATE INDEX, or MODIFY statements.

-no_perm

Does not print GRANT statements.

-no_persist

Does not write CREATE INDEX statements for indexes that have been created with the WITH PERSISTENCE clause.

-no_repmod

Does not write MODIFY table statements for Ingres Replicator system tables of a replicated database.

-no_rep

Does not write Ingres Replicator objects (tables, indexes, events, procedures) of a replicated database to the copy.in file.

-online

Adds the WITH CONCURRENT UPDATES option to the MODIFY statement, if specified.

tablename|viewname

Specifies the tables to be copied. If omitted, all tables are copied. This could also be a list of views; in that case only the given views are copied.

The table name can be qualified with a valid schema name in the format *schema*.*tablename*, as described in Schema Qualifier (see page 18).

Note: No more than 100 objects can be specified. This limit can be raised by modifying the utexe.def file. For more information, see the *Database Administrator Guide*.

-help

Displays command syntax online.

copydb Example on Windows

The following commands make a copy of olddb. In this example, replace the named directory (\mydir\backup) with your own:

```
cd \mydir\backup
copydb olddb
sql olddb<copy.out
```

The following example creates a new database newdb:

```
createdb newdb
sqlnewdb<copy.in
sysmod newdb
```

The following command runs copydb with parameters supplied in a file called flagfile:

```
copydb -param_file=flagfile
```

where flagfile can contain the following entries:

```
dbname
-order_ccm
-relpath
-no_loc
-all
```

This is equivalent to the command:

```
copydb dbname -order_ccm -relpath -no_loc -all
```

copydb Example on UNIX

These commands copy mydb to tape. In this example, replace the named directory (/usr/mydir/backup) with your own:

```
cd /usr/mydir/backup
copydb mydb /usr/mydir/backup
sql mydb <copy.out
tar c
rm *
```

The following commands copy a tape to mydb. In this example, replace the named directory (/usr/mydir/backup) with your own:

```
cd /usr/mydir/backup
tar xvpf /dev/rmt0
sql mydb <copy.in
sysmod mydb
```

copydb Example on VMS

These commands make a static copy of olddb. In this example, replace the named default directory (mydir.backup) with your own:

```
createdb newdb
set default [mydir.backup]
copydb olddb
sql olddb <copy.out
```

Next, the following commands make a copy backup of olddb into newdb. In this example, replace the named default directory (mydir.backup) with your own:

```
set default [mydir.backup]
sql newdb <copy.in
sysmod newdb
```

copyform Command—Copy a Form to Another Database

Copyform is a Visual-Forms-Editor command that copies a form, a QBFName, or a JoinDef from one database to another.

The process has two steps:

1. Copy one or more forms, QBFNames, or JoinDefs from a database to a text file by using the first syntax below.
2. Copy the objects from the text file into a database by using the –I flag shown in the second syntax below.

One type of object only (form, QBFName, JoinDef) can be specified in a single copyform command.

Because the process has two steps, you can use copyform to change the ownership of a form, QBFName, or JoinDef. Copy the desired object into a text file, then copy the form back into the database under a new owner.

For a discussion on using the copyform command to change ownership of a database, see the *Database Administrator Guide*.

The copyform command has the following formats.

Copy the object from the database to a text file:

```
copyform dbname |vnode::dbname[/server_class] form {form} | -q qbfname
{qbfname} | -j joindef {joindef} filename [-s] [-uusername] [-Ggroupid]
```

Copy the object from the text file to the database:

```
copyform -i dbname |vnode::dbname[/server_class]  [-r] filename [-s] [-
uusername] [-Ggroupid]
```

dbname

Specifies the name of the database containing the forms, or to which the forms are being copied. Also specifies the *vnode* and *server_class*, if required, as described in Standard Flags and Parameters (see page 13).

form

Specifies the name of the form. No QBFNames or JoinDefs associated with the form are copied.

Note: No more than 100 objects can be specified. This limit can be raised by modifying the utexe.def file. For more information, see the *Database Administrator Guide*.

Copyform copies only the form's definition to a text file. The format of this text file is not useful for determining what is in the form. Most of the file consists of the entries from the system catalogs in text format. Because these catalogs are interdependent, do not alter or edit this file, or you can cause the irrecoverable destruction of the form.

-q *qbfname*

Copies the specified QBFName. Multiple QBFNames can be specified. Forms and JoinDefs associated with the *qbfname* are also copied. The space between –q and *qbfname* is required.

-j *joindef*

Copies the specified JoinDef. Multiple JoinDefs can be specified. The space between –j and *joindef* is required.

filename

Specifies the name of a text file in which to write the forms, or the name of the text file previously created by copyform, that contain the forms to be copied into the database.

-s

Suppresses status messages.

-i

Indicates this is an input operation. This parameter is required for the input step.

-r

Suppresses the verification prompt for overwriting existing objects. If an object exists in the database under the same name and owner, the object from the file overwrites it. If this flag is not specified, the user is prompted for verification.

-u*username*

Copies forms owned by the specified user, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

copyform Examples

In the following examples, if your database resides on a remote system—for instance, the projects database on the hq node—specify the database as hq::projects rather than projects.

1. The following command line copies out a form called employees from the projects database into a text file called empform.txt:

   ```
   copyform projects empform.txt employees
   ```

2. The following command line copies out a JoinDef called emp_join from the projects database into the empinfo.txt file:

   ```
   copyform projects empinfo.txt -j emp_join
   ```

3. To force copyform to prompt you for a list of QBFNames from the projects database, use the following command line:

   ```
   copyform projects empdata.txt -q
   ```

4. The following command copies the text file empform.txt into the newemp database:

   ```
   copyform -i newemp empform.txt
   ```

copyrep Command—Copy a Report to a Text File

The copyrep command copies a report specification from a database to a text file. The reports must have been created by either Report-By-Forms (RBF) or Report-Writer.

Copying a report into a new database is a two-step process:

1. Copy one or more reports from a database to a text file using the copyrep command.
2. Copy the reports from the text file into a database using the sreport command.

You can use the copyrep and sreport commands to copy a report from one database to another. You can use also these commands to change ownership of a report. To do this, copy out a report owned by a particular user into a text file and then copy the report back into the database under the ownership of another user.

For a discussion on using the copyrep and sreport commands to change ownership of a report, see the *Database Administrator Guide*.

The copyrep command has the following format:

```
copyrep dbname |vnode::dbname[/server_class] filename reportname {reportname}
[-f] [-s] [-uusername] [-Ggroupid]
```

dbname

Specifies the name of the database containing the reports to be copied. Also specifies the *vnode* and *server_class*, if required, as described in Standard Flags and Parameters (see page 13).

filename

Specifies the name of a text file in which to write the report definitions.

reportname

Specifies the name of the report to be copied.

-f

Writes the reports out in the same format as the Archive operation accessed through the Reports Catalog frame of RBF. For reports created with RBF, this strips out many of the statements.

-s

Suppresses status messages.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

copyrep Example

The following example copies a report called emphours in the emp database into a text file called emphours.rw:

```
copyrep emp emphours emphours.rw
```

Then you can use the sreport command to copy the report in the text file emphours.rw into the database newemp under a different owner:

```
sreport newemp emphours.rw -uuser2
```

Note: You must be the DBA for the newemp database or have the security privilege to use the -u flag.

createdb Command—Create a Database

The createdb command creates a new database. The user who creates a database becomes the DBA for that database.

By default, all users have access to a database although access to tables in the database must be explicitly granted. To create a private database, use the -p flag.

Before you can specify file locations in the createdb command, the directories must exist. If not specified, a default location, created during installation, is assumed. For procedures on creating alternate locations, see the *Database Administrator Guide*.

Note: When a database is created, system catalogs are created with the server default page size, unless specified differently on the –page_size parameter.

Note: If createdb fails for any reason, destroy the partially created database using the destroydb command.

Note: If the character set for the installation is selected as UTF8, by default the createdb command creates a Unicode-enabled database with Normalization Form C.

The createdb command has the following format:

```
createdb dbname[/server_class] [cdbname] [-dlocationname] [-clocationname]
[-jlocationname] [-blocationname] [-wlocationname] [-f product {product}]
[-llanguage] [-n[collationname]] [-i[collationname]][-p] [-S][-uusername]
[-rlocationname] [-page_size=n]
```

dbname

Specifies the name of the database to be created. The name must be unique and begin with an alphabetic character. The name can have a maximum of 24 alphanumeric characters (the underscore is also allowed).

Also specifies the *server_class*, if required, as described in Standard Flags and Parameters (see page 13). If you are using Ingres Star, you must specify star as the *server_class*. For examples, see the *Ingres Star User Guide*.

cdbname

Overrides the default coordinator database name stored in the Ingres Star catalogs. The default name of the coordinator database is the *dbname* you specified, prefixed with ii. This is an optional parameter for use with Ingres Star.

-d*locationname*

Specifies the location of the database files. The default is the location to which II_DATABASE points.

-c*locationname*

Specifies the location of the checkpoint files. The default is the location to which II_CHECKPOINT points.

-j*locationname*

Specifies the location of the journal files. The default is the location to which II_JOURNAL points.

-b*locationname*

Specifies the location of the dump files. The default is the location to which II_DUMP points.

-w*locationname*

Specifies the location of the work files. The default is the location to which II_WORK points.

-f *product*

Specifies user interface products for which you want to create catalogs. Valid product names are ingres, ingres/dbd, vision, windows_4gl, and nofeclients, as described in Standard Flags and Parameters (see page 13). The default is to include all product names.

-l*language*

Specifies the collating sequence for a non-Unicode-enabled database. The sequence must exist in the installation before issuing the createdb command.

A database's collating sequence determines the order in which data is sorted.

Valid values for *language* are:

multi

DEC Multinational Character Sequence

spanish

Spanish alphabet's character sequence

collation_name

A custom collation sequence. For details see the *System Administrator Guide*.

To view the available collation sequences, examine the contents of the collation file (located in $II_SYSTEM/ingres/files/collation).

If the -l flag is not specified, the collating sequence is determined by the value of II_COLLATION (if this is set). If II_COLLATION is not set, the default collating sequence is assigned to the database. The default is the native sequence of the ASCII or EBCDIC character set, depending on which is present in your computer.

-n[*collationname*]

Creates a Unicode-enabled database using Normalization Form D (NFD). This enables storing and manipulating Unicode data by defining columns as Unicode data types (that is, nchar, nvarchar, and long nvarchar). The database uses NFD for normalization of Unicode strings for processing and storage. NFD results from the canonical decomposition of a Unicode string.

Collation name can be one of the following:

udefault

(Default) Creates a Unicode-enabled database with the default collation sequence.

unicode_french

Creates a Unicode-enabled database with French collation sequence. For example:

```
createdb -nunicode_french unidb
```

-i[*collationname*]

Creates a Unicode-enabled database using Normalization Form C (NFC). This enables storing and manipulating Unicode data by defining columns as Unicode data types (that is, nchar, nvarchar, and long nvarchar). The database uses NFC for normalization of Unicode strings for processing and storage. NFC results from the canonical decomposition of a Unicode string, followed by the replacement of all decomposed sequences by primary composites, where possible.

Collation name can be one of the following:

udefault

(Default) Creates a Unicode-enabled database with the default collation sequence.

unicode_french

Creates a Unicode-enabled database with French collation sequence. For example:

```
createdb -iunicode_french unidb
```

-p

Creates a private database. Only the DBA and names specified on the accessdb command have access to the database. Do not use this flag with Ingres Star.

-S

Creates an iidbdb. You must be a privileged user to use this flag. Do not use it with Ingres Star.

On VMS, enclose this flag in double quotation marks ("-S").

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-r*locationname*

Specifies the new location of the read-only database; typically this will be the CD-ROM drive where the read-only database is located.

-page_size=*n*

Creates a database with catalogs that have the specified non-default page size.

Limits: Possible values for *n* are: 2048, 4096, 8192, 16384, and 65536.

Example: createdb *dbname* –page_size=4096

Note: The -dmf_cache_size parameter for the DBMS Server should be enabled for the page size specified in this command.

createdb Examples

This command creates a private database on the default devices:

```
createdb -p mydb
```

This command creates the public database ericsdb using a different user name:

```
createdb ericsdb -ueric
```

This command creates a database with its database, checkpoint, and journal files on different devices:

```
createdb bigdb -ddb_ingres -cnewdev_ingres -jotherdev_ingres
```

This command creates a database with catalogs for Ingres and OpenROAD:

```
createdb testdb -f ingres windows_4gl
```

This command creates a distributed database for use with Ingres Star:

```
createdb mydb/STAR
```

createdb Example: Create Unicode-enabled Database

This command creates a Unicode-enabled database with default collation sequence and supports Normalization Form D:

```
createdb -n unicodedb
```

This command creates a Unicode-enabled database with a custom collation sequence and supports Normalization Form D:

```
createdb -nmyunicollation unicodedb
```

This command creates a Unicode-enabled database with default collation sequence and supports Normalization Form C:

```
createdb -i unicodedb
```

This command creates a Unicode-enabled database with a custom collation sequence and supports Normalization Form C:

```
createdb -imyunicollation unicodedb
```

cscleanup Command—Deallocate Shared Memory

Valid on UNIX.

Permission required: Installation owner.

The cscleanup utility deallocates the UNIX shared memory and semaphore resources that were allocated by csinstall for use by Ingres. Typically, you will have no need to run this utility because it is called by the ingstop command.

Use cscleanup when an Ingres server has aborted or you are forced to stop a servers by using the UNIX kill command. To verify the cleanup, use the csreport command and the UNIX command ipcs. If this utility fails, you can remove Ingres shared memory and semaphores with the UNIX command ipcrm.

The cscleanup utility does *not* deallocate shared memory buffer caches. For information about destroying shared caches, see cacheutil Command (see page 40).

The cscleanup command has the following format:

```
cscleanup
```

csreport Command—Display Shared Memory Information

Valid on UNIX.

Permission required: Installation owner.

The csreport utility displays shared memory and semaphore information for your installation.

Verify that the installation owner is the owner of any shared memory segments or semaphores used by your Ingres installation. Running csinstall as root or some other user can cause segments and semaphores to be allocated that are **not** owned by the installation owner. Cscleanup does **not** remove these, so they must be removed under the user login of the user who created them. To do this, use the UNIX utility ipcrm in the format: ipcrm -m*mid* | -s*sid* , where *mid* is the ID number of the shared memory segment and *sid* is the semaphore identifier.

The csreport command has the following format:

```
csreport
```

csreport Output

Here is sample output from the csreport command:

```
!Installation version 610008
!Max number of servers 16
!Description of shared memory for control system:
!Key 0x493D183C: size 16384 attach 00000000
!Description of shared memory for logging & locking system:
!Key 0x493D184E: size 229376 attach C0005000
!Semaphore information for installation:
!  sysV semid 38, num sems 21, used sems 19
!0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
!Event system: used space 13268, length space 16384
```

csreport Example: Show Server Connect IDs

The following command shows the processes that are currently running. You can use this command to obtain the server connect IDs for iimonitor input.

```
csreport | grep "inuse 1"
```

Here is sample output:

```
!inuse 1, pid 3375, connect id 2217, id_number 0, semid 3900
!inuse 1, pid 3384, connect id 2220, id_number 1, semid 4101
!inuse 1, pid 3389, connect id , id_number 2, semid 0
```

The Recovery Server is listed first and can be connected to iimonitor with the connect ID 2217. The other server in this listing has the connect ID 2220. Processes (as opposed to servers) do not have a connect ID.

dclgen Command—Generate Structure Declarations

The dclgen command generates a structure declaration file from table descriptions. The file contains a structure corresponding to the database table, and a declare table statement that serves as a comment and identifies the database table and columns from which the structure was generated.

When the file is generated, use an embedded SQL include statement to incorporate it into the variable declaration section. See the example that follows.

The dclgen command has the following format:

```
dclgen [language] [dbname] [tablename] [filename] [structurename]
```

language

Specifies the embedded SQL host language, such as C.

dbname

Specifies the name of the database containing the table.

tablename

Specifies the name of the database table.

filename

Specifies the name of the output file into which the structure declaration is placed.

structurename

Specifies the name of the host language structure that the command generates. The structure tag is the structure name followed by an underscore character (_).

dclgen Example

The following example demonstrates how to use DCLGEN within a C program (assuming the Employee table was created in the Personnel database):

```
exec sql create table employee
       (eno      smallint not null,
        ename     char(20) not null,
        age       integer1,
        job       smallint,
        sal       decimal(14,2) not null,
        dept      smallint)
        with  journaling;
```

The DCLGEN system-level command is:

```
dclgen c personnel employee employee.dcl emprec
```

This command creates the employee.dcl file, which contains a comment and two statements. The first statement is the declare table description of employee, which serves as a comment. The second statement is a declaration of the C structure emprec.

The contents of the employee.dcl file are:

```
/* Table employee description from database personnel */
exec sql declare employee table
               (eno        smallint not null,
                ename       char(20) not null,
                age         integer1,
                job         smallint,
                sal         decimal(14,2) not null,
                dept        smallint);

struct emprec_ {
                short     eno;
                char      ename[21];
                short     age;
                short     job;
                double    sal;
                short     dept;
} emprec;
```

The length of the ename buffer is increased by one byte to accommodate the C null terminator. Also, the integer1 data type is mapped to short rather than char.

To include this file in an embedded SQL declaration section, use the embedded SQL include statement:

```
exec sql begin declare section;
        exec sql include 'employee.dcl';
exec sql end declare section;
```

You can then use the emprec structure within a select, fetch, or insert statement.

The field names in the structure that DCLGEN generates are identical to the column names in the specified table. Therefore, if the column names in the table contain any characters that are illegal for host language variable names you must modify the name of the field before attempting to use the variable in an application.

delobj Command–Delete Objects from Database

The delobj command deletes objects from the database.

For more information on using the delobj command to delete specific object types, see the *Character-based Querying and Reporting Tools User Guide*.

The delobj command has the following format:

```
delobj dbname|vnode::dbname[/server_class] objectspec | all [-silent]
[-uusername] [-Ggroupid]
```

dbname

Specifies the name of the database and, if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

objectspec

Describes the object or objects to be deleted from the database. Either an objectspec or the word -all can be specified. You can optionally specify the silent -u and -G flags.

Details on specifying this parameter are described in Object Specification (see page 77).

-all

Specifies that all objects owned by the effective user are to be deleted. You can optionally specify the silent -u and -G flags.

-silent

Runs in silent mode, suppressing status messages.

-uusername

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-Ggroupid

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose the –G*groupid* parameter in double quotes.

Object Specification for delobj Command—Specify Objects to Delete

The object specification names one or more objects to be deleted from the database using the delobj command. You can specify one type of object only.

The *objectspec* parameter has the following format:

```
-report|-form|-joindef|-graph|-application|-qbfname {objname} [-wildcard]
[-include filename]
```

-report

Indicates that the object names refer to reports.

-form

Indicates that the object names refer to forms.

-joindef

Indicates that the object names refer to join definitions.

-application

Indicates that the object names refer to applications.

-qbfname

Indicates that the object names refer to QBFNames.

objname

Specifies one or more object names of the specified type.

An object name can contain SQL wildcard characters (%, _, [ ]), with the backslash (\) as the escape character. For details, see the pattern matching description in the *SQL Reference Guide*.

-wildcard

Expands wild cards. The default is not to expand wild cards.

-include *filename*

Specifies the name of an ASCII file that contains object names. This parameter is an alternative to listing multiple *objname* parameters on the command line.

The file can consist of one or more ASCII lines with the following format characteristics:

- Each line can be up to 256 bytes (not characters) in length. Longer lines may cause errors.
- A line can contain a combination of object names, comment, and white space (blanks and tabs).
- A comment begins with the character # and continues to the end of the line. Each line terminates with an ASCII carriage return (CR), line feed (LF), or form feed (FF) character.

delobj Examples

1. This command deletes the empreport report from the empdata database:

   ```
   delobj empdata -report empreport
   ```

2. This command deletes the videoapp application from the video database, specifying user name Adder:

   ```
   delobj video -uAdder -application videoapp
   ```

3. This command deletes the forms listed in a prepared namefile:

   ```
   delobj video -form -include namefile
   ```

   The namefile contains the following:

   ```
   wa1 wa2 wa3 wa4 wa5 wa6 wa7 wa8 wa9 wa10 wa11 wa12 wa13
   wr1 wr2 wr3 wr4 wr5 wr6 wr7 wr8 wr9 wr11 wr12 wr13
   wr14 wr15 wr16 #Includes all wr names except wr10.
   ```

4. In the following examples, assume objects: qbfname_1, qbfnameA1, and qbfnameB1.

   a. The following command deletes all three objects using the underscore as a wildcard character:

      ```
      delobj dbname -qbfname qbfname_1 -wildcard
      ```

   b. The previous command without the **-**wildcard flag deletes the qbfname qbfname_1 only:

      ```
      delobj dbname -qbfname qbfname_1
      ```

   c. The previous command is identical to the following command, which escapes the underscore character, preventing it from being used as a wildcard character, even though the **-**wildcard flag is specified:

      ```
      delobj dbname -qbfname qbfname\_1 -wildcard
      ```

5. Deletes all reports listed in the oldrpts.txt file from the admin database:

   ```
   delobj admin -report -include oldrpts.txt
   ```

   Some examples of possible entries in the oldrpts.txt file are:

   ```
   # Example of a comment in an included file
   oldrpt           # Comment following a report name
   oldrpt           # Comment closely following a report name
   sales\_old       # Deletes only the report, sales_old
   sales_1          # Deletes salesA1, sales B1, and so on
   sales%           # Deletes reports beginning with "sales"
   sales\[1-9\]     # Deletes reports sales1 through sales9
   proj3 proj7      # Example of reports separated by a space
   rpt1    rpt2     # Example of reports separated by a tab
   ```

dereplic Command—Remove Objects from Replicated Database

The dereplic command removes the Ingres Replicator database objects (queues and tables, events, and database procedures) from a replicated database.

The dereplic command has the following format:

```
dereplic [vnode::] dbname [-udba_name]
```

vnode*:: *dbname

Specifies the name of the database. If required, specify the vnode.

-u*dba_name*

Specifies the effective user for the session. You must run dereplic as the owner of the database.

destroydb Command—Destroy a Database

Permission required: DBA or system administrator.

The destroydb command removes an existing database. The directory of the database and all files in that directory are removed. You cannot destroy the iidbdb using destroydb.

If you are using Ingres Star, the destroydb command destroys the distributed database, the coordinator database, and all the Ingres Star objects that make up the distributed database. Data in underlying tables in non-coordinator local databases registered in the distributed database are not affected.

The destroydb command has the following format:

```
destroydb dbname [-p] [-l] [-uusername]
```

dbname

Specifies the name of the database.

-p

Prompts you to be sure that you want to destroy the database.

VMS: If you want to be prompted for confirmation automatically when you execute the destroydb command, use the following command, which eliminates the need to use the –p flag to obtain a confirmation prompt:

```
destroydb:=="ii_system:[ingres.bin] destroydb.exe -p"
```

-l

Confirms if the database is in use, and if in use, returns with an error message.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

destroydb Examples

This command destroys the empdata database:

```
destroydb empdata
```

This command destroys the video database as the user Brad:

```
destroydb video -uBrad
```

eqc Command—Invoke Embedded QUEL Preprocessor for C

The eqc command invokes the embedded QUEL preprocessor for C.

The eqc command has the following format:

```
eqc {flags} [filename]
```

flags

Specify options to the preprocessor. For details on the flags, see the *Embedded QUEL Companion Guide*.

filename

Indicates the name of the file containing the embedded QUEL C source for your application.

For information on compiling and linking QUEL/C programs that is specific to your operating system, check the Readme file.

eqc Examples

This command preprocesses file1.qc to file1.c:

```
eqc file1
```

This command preprocesses file2.xc to file2.c and creates listing file2.lis:

```
eqc -l file2.xc
```

This command accepts input from standard input and writes generated code to standard output:

```
eqc -s
```

This command preprocesses file3.qc to file3.out:

```
eqc -ffile3.out file3
```

This command displays a list of flags available for this command:

```
eqc
```

This command preprocesses and compiles the file test1:

```
eqc test1.qc
cc -c test1.c
```

esqla Command—Invoke Embedded SQL Preprocessor for Ada

The esqla command invokes the Ingres embedded SQL (ESQL) preprocessor for Ada.

The esqla command has the following format:

```
esqla [flags] [filename]
```

flags

Specify options to the preprocessor.

These flags are common to most host language preprocessors, and are described under the esqlc command (see page 84).

The following flag is unique to the preprocessor for Ada:

-avads

On VMS, generates code for VADS on systems that have both an OpenVMS preprocessor and VADS Ada preprocessor. This flag is required.

filename

Specifies the name of the file that contains the embedded SQL statements.

For a complete description of the SQL preprocessor requirements for Ada, see the *Embedded SQL Companion Guide*.

esqlb Command—Invoke Embedded SQL Preprocessor for BASIC

The esqlb command invokes the Ingres embedded SQL (ESQL) preprocessor for BASIC.

The esqlb command has the following format:

```
esqlb [flags] [filename]
```

flags

Specify options to the preprocessor.

These flags are common to most host language preprocessors, and are described under the esqlc command (see page 84).

The following flags are unique to the preprocessor for BASIC:

-i*N*

Sets the default size of integers to *N* bytes. *N* must be 1, 2, or 4. The default is 4.

-r*N*

Sets default size of reals to *N* bytes. *N* must be 4 or 8. The default is 4.

filename

Specifies the name of the file that contains the embedded SQL statements.

For a complete description of the SQL preprocessor requirements for BASIC, see the *Embedded SQL Companion Guide*.

esqlc Command—Invoke Embedded SQL Preprocessor for C

The esqlc command invokes the Ingres embedded SQL (ESQL) preprocessor for C.

The esqlc command has the following format:

`esqlc [`*`flags`*`] [`*`filename`*`]`

filename

Specifies the name of the file that contains the embedded SQL statements.

flags

Specify options to the preprocessor.

These flags are common to most host language preprocessors.

Note: Flags specific to the preprocessor for C are noted with (esqlc).

-{# | p}

(esqlc) Generates **#** line directives to the C compiler (by default, they are in comments). This flag is helpful when debugging the error messages from the C compiler.

-?

On Windows and VMS, lists the valid command line options.

--

On UNIX, lists the valid command line options.

-[**no**]**blank_pad**

(esqlc) Tells the preprocessor to pad (**-**blank_pad) or not pad (-noblank_pad) with blanks any data selected at run time into fixed-length char host variables. Padding is done to the declared length of the variable, less one byte for the C null terminator.

The setting -blank_pad generates code that complies with ANSI and ISO Entry SQL-92 data retrieval rules for fixed-length char variables.

The setting -noblank_pad complies with current data retrieval rules. Instead of padding with blanks, char data is null terminated to the length of the data retrieved. The default is -noblank_pad.

-[no]check_eos

(esqlc) Tells the preprocessor to check (check_eos) or not check (-nocheck_eos) for an end-of-string null terminator on char variables.

The setting -check_eos is provided for ANSI SQL-92 conformity. It raises an error if a null terminator is missing.

The setting -nosqlcode is the default. It turns off the above checking.

-d

Adds debugging information to the runtime database error messages generated by ESQL. The source file name, line number, and the erroneous statement are printed along with the error message.

-f[*filename*]

Writes preprocessor output to the specified file.

If the filename variable is omitted, the output is sent to standard output, one screen at a time.

If the -f flag is omitted, output is written to a file that has the same base name as the input file, and contains an extension corresponding to the language preprocessor you invoked. For information about filename extensions, consult your host language companion guide.

-i*N*

(esqlc) Sets the default size of integers to *N* bytes. *N* must be 1, 2, or 4. The default is 4.

-l

Writes preprocessor error messages to the preprocessor's listing file and the terminal. The listing file includes preprocessor error messages and your source text in a file named *filename*.lis, where filename is the name of the input file.

-lo

Makes generated code appear in the listing file too.

-multi

Generates a thread safe code for use in multi-threaded ESQL applications.

-o[**.** *ext*]

Specifies whether to generate output files for include files.

If no extension is specified, the include file preprocessor does not create an output file. This does not affect the inclusion of files in the main program. The preprocessor generates a default extension for the translated include file statements unless you specify the -o.*ext* flag.

If an extension is specified, the include files are output to the specified *extension* after being preprocessed.

-s

Reads embedded commands from standard input and generates resulting code to standard output. This is useful for unfamiliar testing statements. If the -l option is specified with this flag, the listing file is called stdin.lis.

On Windows, to terminate the interactive session, type Ctrl + C.

On UNIX, to terminate the interactive session, type Ctrl + D.

On VMS, to terminate the interactive session, type Ctrl + Z.

-[no]sqlcode

Tells the preprocessor to assume (-sqlcode) or not assume (**-**nosqlcode) the existence of a status variable named SQLCODE to receive status information from SQL statements. The declaration does not have to be in an exec sql begin/end declare section.

The -sqlcode setting is provided for ANSI SQL-92 conformity.

The -nocheck_eos setting is the default.

-w

Prints warning messages.

-wsql=entry_SQL92 | **open**

Issues a warning if the preprocessor detects an embedded SQL statement that does not follow the specified syntax.

entry_SQL92 specifies the ANSI.SQL-92 entry level standard. (This is also known as the FIPS flagger option.)

open specifies OpenSQL syntax. This flag is useful if you intend to port an application across different Enterprise Access products. Warnings do not halt or affect the success of compilation. This flag does not validate the statement syntax for any SQL gateway whose syntax is more restrictive than that of OpenSQL.

Note: The flag –wopen is identical to –wsql=open, and is supported for backward compatibility.

For a complete description of the SQL preprocessor requirements for C, see the *Embedded SQL Companion Guide*.

esqlcbl Command—Invoke Embedded SQL Preprocessor for COBOL

The esqlcbl command invokes the Ingres embedded SQL (ESQL) preprocessor for COBOL.

The esqlcbl command has the following format:

```
esqlcbl [flags] [filename]
```

flags

Specify options to the preprocessor.

These flags are common to most host language preprocessors, and are described under the esqlc command (see page 84).

The following flags are unique to the preprocessor for COBOL:

-a

Generates output in ANSI format. Use this flag if your source code is in ANSI format and you want to compile the program with the COBOL command line qualifier ansi_format.

On VMS, if this flag is omitted, the preprocessor generates output in Compaq COBOL terminal format.

filename

Specifies the name of the file that contains the embedded SQL statements.

For a complete description of the SQL preprocessor requirements for COBOL, see the *Embedded SQL Companion Guide*.

esqlcc Command—Invoke Embedded SQL Preprocessor for C++

The esqlcc command invokes the Ingres embedded SQL (ESQL) preprocessor for C++.

The esqlcc command has the following format:

```
esqlcc [flags] [filename]
```

flags

Specify options to the preprocessor.

These flags are common to most host language preprocessors, and are described under the esqlc command (see page 84).

The following flags are unique to the preprocessor for C++:

-extention=*ext*

Specifies the extension for the C++ file created by the precompiler.

-prototypes

Directs the preprocessor to include a header file containing ANSI style function prototypes for the Ingres run time routines. The default is -noprototypes (the prototypes in the header file are not ANSI style).

filename

Specifies the name of the file that contains the embedded SQL statements.

For a complete description of the SQL preprocessor requirements for C++, see the *Embedded SQL Companion Guide*.

esqlf Command—Invoke Embedded SQL Preprocessor for Fortran

The esqlf command invokes the Ingres embedded SQL (ESQL) preprocessor for Fortran.

The esqlf command has the following format:

```
esqlf [flags] [filename]
```

flags

Specify options to the preprocessor.

These flags are common to most host language preprocessors, and are described under the esqlc command (see page 84).

The following flags are unique to the preprocessor for Fortran:

-i*N*

Sets the default size of integers to *N* bytes. *N* must be either 2 or 4. The default is 4.

On UNIX, if *N*=2 is used, the -i2 flag must be specified.

On VMS, if *N*=2 is used, the noi4 qualifier must be used.

filename

Specifies the name of the file that contains the embedded SQL statements.

For a complete description of the SQL preprocessor requirements for Fortran, see the *Embedded SQL Companion Guide*.

extenddb Command—Extend Database to New Location

The extenddb command extends databases to new or existing locations.

The extenddb command has the following format:

```
extenddb -llocation [-uuser] [dbname...|-nodb] [-aarea_dir]
[-Udata,ckp,jnl,dmp,work|awork] [-rraw_pct] [-drop] [-alter]
```

-l*locname*

Identifies the name of the new or existing location.

-u*user*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

***dbname*...|-nodb**

Specifies the list of databases to be extended. If no databases are to be extended, use the –nodb flag. This flag allows for the creation of a location without extending databases to it.

Also specifies the *vnode*, if required, as described in Standard Flags and Parameters (see page 13).

-a*area_dir*

Specifies the directory (*area_dir*) that the new location will point to. This option creates the full directory path below the ingres root location. Use this option when creating a new location only. Do not use it when extending a database to an existing location.

Note: On VMS, this option does not create the area.

-Udata,ckp,jnl,dmp,work|awork

Specifies the usage for the new location. Valid usages include data (database), ckp (checkpoint), jnl (journal), dmp (dump), work (work), and (awork) auxiliary work.

-r*raw_pct*

Specifies, for Raw locations, the percentage of the Raw Area to be allocated to this location. Use this option when creating a new location only. Do not use it when extending a database to an existing location.

-drop

Drops the specified location.

-alter

Modifies a location's usage to add data, ckp, jnl, dmp, or work areas.

extenddb Examples

This command extends the stockdb database to use the directory /disk1/loc1 as new data and work areas:

```
extenddb -lextraloc1 stockdb -a/disk1/loc1 -Udata,work
```

This command creates a location without extending databases to it:

```
extenddb -lextraloc2 -a/disk2/loc2 -Uckp,jnl -nodb
```

This command extends the employeedb database to an existing location:

```
extenddb -lextraloc3 -Udata employeedb
```

fastload Command—Load Binary Files into Database

The fastload command loads binary format files into the database.

Note: Before using this command, make sure you understand the fastload operation and have satisfied the requirements for running it. For more information, see the *Database Administrator Guide*.

The fastload command has the following format:

```
fastload dbname -file=filename -table=tablename
```

dbname

Identifies the name of the database into which the file is to be loaded.

file=*filename*

Identifies the file name to be loaded. If the file is not in the current directory, specify the full path name of the file.

-table=*tablename*

Identifies the table into which the file is to be loaded.

fastload Example

This command loads the file test.out into the table test in the database fload:

```
fastload fload -file=test.out -table=test
```

genxml Command—Export Tables Into XML Format

The genxml utility allows bulk transfer of Ingres data from an Ingres database into XML format. This allows Ingres data on the web to communicate with other third party products.

The generated XML documents refer to generic Ingres DTD (Document Type Definition). Ingres DTD is a controlled document describing the Ingres data. DTD can be internal or external.

When the genxml command is executed on an Ingres database, an XML file is generated, containing the metadata and the data for the tables in XML format, as specified by the Ingres DTD (Document Type Definition). A DTD file is also generated by genxml by default, unless other options are specified.

For external DTDs, the dtd file can be printed in the same directory as the XML file. Alternatively, external DTDs can be a referenced in the XML file to the DTD location. This location will be either a URL or a static location, such as $II_SYSTEM/ingres/files/ingres.dtd. Reference to the Ingres DTD is made using the DOCTYPE declaration in the generated XML file.

You can use the generated XML file as input to the xmlimport utility for importing into another Ingres installation. For details, see xmlimport Command (see page 286).

The genxml command has the following format:

```
genxml dbname | vnode::dbname[/server_class] [-uuser] [-P] [-GgroupID]
[-dest=dir] [-xmlfile=filename] [-dtdfile=filename] [-metadata_only]
[-internal_dtd] [-referred_dtd] [{tablename}] [title_doctype=title]
```

dbname

Specifies the name of the database being exported. Also specifies the *vnode* and *server_class*, if required, as described in Standard Flags and Parameters (see page 13).

-u*user*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-P

Specifies the password if the session requires one.

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

-dest=*dir*

Specifies the destination directory into which the XML file is generated. An empty *dirname* specification (".") denotes the current directory. The generated Ingres DTD is also placed in this directory.

-xmlfile=*filename*

Specifies the name of the output XML file. By default, the file is called xmlout.xml.

-dtdfile=*filename*

Specifies the name of the output dtd file. By default, the file is called ingres.dtd.

-metadata_only

Indicates to print the metadata information only.

-internal_dtd

Prints the DTD inline inside the XML doc.

-referred_dtd

Places a reference to the ingres.dtd in Ingres files directory ($II_SYSTEM/ingres/files).

tablename

Specifies the table name or names that the user wants the XML to output. Table names must be separated by spaces. If omitted, all tables are copied. The table name can be qualified with a valid schema name in the format *schema*.*tablename*, as described in Schema Qualifier (see page 18).

Note: No more than 100 objects can be specified. This limit can be raised by modifying the utexe.def file. For more information, see the *Database Administrator Guide*.

-title_doctype=*title*

Changes the doctype or the document name of the XML file. The default doctype is IngresDoc. Use this flag with caution. If the document name is changed, the referred_dtd option should not be used because the referred generic Ingres DTD in $II_SYSTEM/ingres/files has IngresDoc as document type.

genxml Examples

This command generates a copy of a database testdb in XML format:

```
genxml testdb
```

The generated files “xmlout.xml” and “ingres.dtd” are written to the current directory. The file xmlout.xml contains the metadata and data of all tables in the database testdb.

The following command exports table tab1 only, writes the files to directory (/tmp/mydirectory) using file name myxml.xml, and places the DTD file inline with the XML file:

```
genxml testdb -dest=”/tmp/mydirectory” -xmlfile=myfile.xml -internal_dtd tab1
```

If you want to recreate objects at any other location or installation without copying a large amount of data, you can use the -metadata_only flag to generate the metadata information in XML format.

iea Command—Start the Export Assistant

Permission required: Access to the directory where the utility is located.

The iea command invokes the Export Assistant, a graphical user interface that allows you to export data into a file format that is accepted by external products.

The iea command has the following formats:

```
iea [-loop] [filename.ii_exp]

iea {filename.ii_exp|-l listfile} -silent -logfilename=logfile.txt [-o]
```

-loop

Prompts whether you want to perform another export operation after an export is complete.

***filename*.ii_exp**

Invokes the Export Assistant with the parameters that are stored in the specified export file. The extension of the export file should be .ii_exp.

-l *listfile*

In silent mode, performs export operations on multiple export files listed in *listfile*.

–silent –logfilename=*logfile*

Runs Export Assistant in "silent mode" (no prompts appear), and reports errors or output in the specified *logfile*. The –silent option is available only if a *filename*.ii_exp or –l argument is used. If a problem occurs, errors are reported in the specified log file and the program terminates with no prompts.

-o

In silent mode, overwrites the log file to be created, if it already exists.

iea Example

This example exports data in multiple files.

In the example, the filelist.txt file contains the following two lines:

```
c:\temp\exportemployees.ii_exp
c:\temp\exportsales.ii_exp
```

The following command invokes the Export Assistant in silent mode. The Export Assistant then performs two exports that correspond to the parameters in the c:\temp\exportemployees.ii_export and c:\temp\exportsales.ii_export files. Any errors are reported in the c:\temp\erriea.txt file. The files to be created will be overwritten if they already exist.

```
iea -l filelist.txt -silent -logfilename=c:\temp\erriea.txt -o
```

iia Command—Start the Import Assistant

Permission required: Access to the directory where the utility is located.

The iia command invokes the Import Assistant, a graphical user interface that allows you to import data from an external product.

The iia command has the following format:

```
iia [-loop] [filename.ii_imp]
```

-loop

Prompts whether you want to perform another import operation after an import is complete.

***filename*.ii_imp**

Invokes the Import Assistant with the import parameters that are stored in the specified file.

iigenres Command—Generate CONFIG.DAT File

Permission required: Access to the directory where the utility is located.

The iigenres utility generates a default configuration for an Ingres installation. Use this utility to regenerate the Ingres configuration file (CONFIG.DAT) if it is accidentally deleted or corrupted.

The iigenres command has the following format:

```
iigenres [-v] host|rule_map|host rule_map
```

-v

Displays system commentary (verbose mode) to the standard output device during the iigenres operation.

host

Specifies the host for which the configuration should be generated. The *host* or *rule-map* parameter is required. You can also specify both a *host* and a *rule_map*.

rule_map

Specifies the rule map file to use. The rule map file contains a list of the rule system files (CRS extension files) to use when generating the default configuration.

iigenres Example

This command creates a config.dat file for the usajemm system using the default rule map file.

```
iigenres usajemm default.rfm
```

iigetres Command—Get the Value of a Resource

Permission required: Access to the directory where the utility is located.

The iigetres utility looks up a value in the default configuration file (config.dat) and prints it to the standard output device. This utility is used by the installation process and CBF.

The iigetres command has the following format:

```
iigetres name
```

name

Identifies the name of the configuration parameter, as it appears in the config.dat file.

iiinitres Command—Install Parameter into CONFIG.DAT

The iiinitres utility installs a newly defined configuration parameter into an installation's configuration file (CONFIG.DAT). The parameter must be defined in a configuration rule file (.crs file).

The iiinitres utility is for use during version upgrades. The system administrator may use it occasionally to install special-use parameters, or to update configuration files that were incompletely updated by older versions of Ingres.

The iiinitres utility locates the named parameter in the configuration rule system to get the leading components of the full parameter name. The parameter is installed in all matching positions of the configuration. For example, if the parameter is found to be a DBMS configuration parameter from the rule system, it will be installed into all DBMS server configurations in the installation. For each entry added to the configuration, any parameters dependent on the newly added resource are recalculated.

The iiinitres command has the following format:

```
iiinitres [-v] [-keep] resource [rule-map]
```

-v

Displays old and new values as well as all recalculations (verbose mode).

-keep

Keeps the current value of the specified parameter (if it exists) in the configuration file. Without the -k option, the default setting will override the existing setting.

resource

Indicates the name of the parameter to install. Specify the final component of the parameter name only, not the full name.

rule-map

Indicates the name of a rule-map file to use in place of the normal one, default.rfm. The rule-map file must be in the $II_SYSTEM/ingres/files directory.

You can use the rule-map file to include special rules for computing the parameter being installed. For example, the Ingres upgrade process uses this option when calculating a replacement parameter from its old form.

iijdbcprop Command—Generate Sample JDBC Driver Properties File

The iijdbcprop command generates a sample Ingres JDBC driver properties file (iijdbc.properties) for use with the JDBC driver (iijdbc.jar).

During installation, the iijdbcprop command is invoked by the Data Access Server configuration utility to create an iijdbc.properties file under $II_SYSTEM/ingres/files directory. This command can be run at any time.

The iijdbcprop command has the following format:

```
iijdbcprop -a | -t | -c | -f fname | -h
```

-a

Lists all supported driver properties, including tracing properties, to STDOUT.

-t

Lists only tracing properties to STDOUT.

-c

Same as flag –a, but writes to iijdbc.properties file.

-f *fname*

Same as flag -a, but writes to a specified disk file.

-h

Displays syntax online.

iijdbcprop Examples

This command lists only the JDBC driver trace properties in the STDOUT:

```
iijdbcprop -t
```

This command creates an iijdbc.properties file in the $II_SYSTEM/ingres/files directory containing all supported JDBC driver properties, most of which are commented out by the # character:

```
iijdbcprop -c
```

iilink Command—Install User-defined Data Type

Permission required: System administrator.

The iilink utility links (loads) the DBMS Server to include a user-defined data type or function.

The iilink command has the following format:

```
iilink [-dbms] [-standard] [-noudt] [-nosol] [-shared] [-merge] [-lp32]|[-lp64] [-help]
```

-dbms

Builds only iidbms.

-standard

(Default) Uses standard binary names.

-noudt

Does not link user-defined data types and does not prompt for input.

-nosol

Does not link the Spatial Objects Library and does not prompt for input.

-merge

Unconditionally builds iimerge binary.

-shared

(Default on Linux) Builds server from shared libraries.

-lp32

Builds 32-bit server. This is the default on 32-bit platforms (Intel Linux, for example) and 32/64 bit hybrid platforms (SPARC Solaris, for example).

-lp64

Builds 64-bit server. This is the default on 64-bit platforms (Tru64, for example) and 64/32-bit reverse hybrid platforms (AMD64 Linux, for example).

-help

Displays command syntax online.

iimkcluster Command—Convert Ingres Instance to Cluster Node

Permission required: Installation owner.

Converts a non-cluster Ingres instance to the first node in a cluster Ingres instance. This utility is for use with Ingres Cluster Solution. See also iuncluster Command (see page 125) and iisunode Command (see page 125).

The iimkcluster command has the following format:

```
iimkcluster
```

iimklog Command—Generate Transaction Log File

Permission required: Installation owner.

The iimklog command generates an Ingres transaction log file. The command is invoked by CBF and the Configuration Manager.

The current transaction log must be deleted before creating a new one.

The iimklog command has the following format:

```
iimklog
```

iimonitor Command—Administer DBMS, Recovery, and GCF Servers

Permission required: Privileged user (server control privilege).

The iimonitor utility allows a system administrator or other privileged user to perform session and server connection functions for DBMS, recovery, and GCF servers, including:

- Viewing the status of the given server
- Shutting down the given server
- Monitoring or shutting down a particular session of the given server
- Performing other server control functions of the given server

If you are using iimonitor to terminate a session that has an active transaction, the server first rolls back the transaction. The session is not completely removed until the rollback is complete.

The iimonitor command has the following format:

```
iimonitor server_id
```

server_id

Identifies the GCF address of the server.

To obtain the GCF address of a server, use the iinamu utility or the csreport utility (on UNIX). The csreport utility shows DBMS and recovery servers only.

This server ID can also be found in the installation log file (errlog.log).

iimonitor Utility Commands

At the IIMONITOR > prompt, several commands are available.

Help Command

The iimonitor help command lists the available commands in the iimonitor utility for the given server.

The help command has the following format:

```
help
```

Show Server Command

The iimonitor show server command displays information about the given server, including the number of sessions currently active or connected to it, the state of the server, and the CPU usage in terms of quanta used.

The show server command has the following format:

```
show server [listen|shutdown]
```

listen

Displays the server listen state, either OPEN or CLOSED

shutdown

Displays the server shutdown state, either OPEN or PENDING

Show Sessions Command

The iimonitor show sessions command displays a list of active sessions and their current states.

The show sessions command has the following format:

```
show [user]|system|all|admin sessions[formatted|stats]
```

user

Gives information on user sessions. This is the default if no option is specified.

system

Provides information on system sessions.

all

Provides information on user, system, and admin sessions.

admin

Provides information on admin (iimonitor) sessions.

Note: This option is for GCF servers only.

formatted

Shows additional information for each session in a block format.

stats

Displays block (message) I/O counts.

Session ID Format in Command Output

The output of the show sessions command displays the session ID in the following format:

ssssssss:*tt*

ssssssss

Is the session ID

tt

Is the thread ID (for DBMS and recovery servers) or the GCA association ID (for GCF servers).

In the example 8125620:2, the notation :2 is the GCA association ID specific to the given server, when available. This ID may also be shown in error logs and trace output.

Note: The association ID in one server (for example, Name Server) does not necessarily match the association ID in another server (for example, Communications Server).

Output of Show All Sessions Command

This command shows a list of all active sessions, their threads, and current states:

```
show all sessions
```

Possible session states displayed in the output are as follows:

CS_EVENT_WAIT:

Indicates the session is waiting for an event. The event type is shown in parentheses, and can be any of these:

(LOCK)—The session is waiting for a lock to be granted.

(DIO)—The session is waiting for a disk I/O to complete.

(LOG-IO) – The session is waiting for the completion of I/O to the transaction log.

(BIO)—The session is waiting for a message to be received from or sent to its associated user interface.

(GWFIO)—The session is waiting for completion of a request it has made through a gateway to a non-Ingres database.

CS_MUTEX

Indicates the session is waiting for a semaphore (access to a system data structure).

CS_COMPUTABLE

Indicates the session is able and waiting to run.

CS_INTERRUPT

Indicates the session's current wait state can be interrupted, if needed.

The system sessions include server threads. Server threads are as follows:

Admin thread

Assists in administrative chores.

Note: This thread cannot be seen with iimonitor.

Idle thread

Assists in administrative chores.

Event thread

Handles event processing.

Write behind thread

Performs write behind processing.

Consistency point thread

Performs consistency points. (This thread was previously called Fast Commit, but all servers—including non-Fast Commit servers-- now use this thread.)

Dead process thread

Checks for abnormal process termination.

Force abort thread

Performs force abort processing.

Group commit thread

Performs group commit processing.

Lock callback thread

Performs all lock callback actions.

Log writer thread

Performs transaction logfile writes.

Security audit thread

Performs security auditing (in C2 enabled servers only).

The iimonitor utility can also be used to connect to the recovery process (DMFRCP). Formatting the recovery thread in the recovery process displays the current state of online recovery operations, if any are taking place. The recovery process is multi-threaded, and has the following threads that can be viewed with iimonitor:

Recovery thread

Performs online recoveries

Consistency point timer thread

Performs timed consistency points

Format Sessions Command

The iimonitor format sessions command displays information about the session. Its output is similar to the output of the show sessions formatted command.

The format sessions command has the following format.

```
format [user]|admin|system|all [sessions]
```

user

Provides information on user sessions. This is the default if no option is specified.

admin

Provides information on admin (iimonitor) sessions.

Note: This option is for GCF servers only.

system

Provides information on system sessions.

all

Provides information on all active sessions.

Set Server Command

The iimonitor set server command controls the state of the given server. This command can be run by a privileged user only.

The set server command has the following format:

```
set server shut|closed|open
```

shut

Disallows additional connections and shuts down the server when currently connected sessions finish.

closed

Disallows additional connections.

open

Allows new connections and cancels a pending set server shut

Stop Server Command

The iimonitor stop server command stops the given server immediately. Client sessions are dropped. This command can be run by a privileged user only. Use this command only if absolutely necessary, for example, if an Ingres tool program is hanging.

The stop server command has the following format:

```
stop server [force]
```

force

Terminates server immediately.

Note: This option applies to GCF servers only.

iimonitor Commands That Perform Actions on Sessions

The following iimonitor commands perform actions on all server sessions or a specific server session.

The iimonitor commands that perform actions on sessions have the following format:

```
remove|suspend|resume|kill session_id
```

remove

Disconnects a particular user session. It cannot be used to drop system threads. This command can be run by a privileged user only.

suspend

Suspends a compute-bound session to allow a trace of the problem. This option is not available for GCF servers.

resume

Resumes a suspended session. This option is not available for GCF servers.

kill

Terminates the currently executing query while leaving the user session connected. This option is not available for GCF servers.

session_id

Specifies to perform the action on the specified session (*sessionid*). To display the session ID, use the iimonitor utility **show sessions** command.

Quit Command

The iimonitor quit command terminates the iimonitor session.

The quit command has the following format:

```
quit
```

iimonitor Commands Specific to GCF Servers

The iimonitor commands that are specific to GCF servers include the following:

- set trace
- register server
- remove tickets
- remove pooled sessions

Set Trace Command

The iimonitor set trace command dynamically enables or disables tracing on a GCF server. Dynamic tracing allows tracing to be limited to specific times, which can be useful to trap specific conditions or to reduce the volume of trace output. Dynamic tracing allows tracing to be turned on and off without having to recycle GCF servers.

Note: This command overrides trace settings in config.dat and environment variables.

The set trace command has the following format:

```
set trace attribute value
```

attribute value

Specifies the trace attribute and its value. Valid values depend on the trace attribute being set. Valid attributes are:

level, GCA, GCS, or API (iigcd only)

Specifies type of tracing to perform. “level” turns on the default tracing associated with that GCF server:

GCC—Communications Server - COMSVR (iigcc)

GCD—Data Access Server - DASVR (iigcd)

GCN—Name Server - NMSVR (iigcn)

GCA, GCS or API (for iigcd only) can be specified in addition to or instead of "level" to provide tracing for those areas in the server.

Note: The 3-character trace mnemonic is equivalent to setting Ingres variable II_XXX_TRACE or config.dat XXX_trace_level for the server connected to, where XXX is GCA, GCS, API, GCC, GCD, or GCN. For more details on tracing GCF servers for diagnostic purposes, see the *Connectivity Guide* and *System Administrator Guide*.

The associated attribute value is the level of tracing from 0-5, with a higher value providing more detailed tracing.

log

Specifies a trace output log file. The associated attribute value is the full disk path and file name of the log file or NONE to disable all tracing.

Note: All tracing from a server is directed to a single output file. As a result, any trace settings configured using a non-dynamic method (that is, configured through config.dat instead of iimonitor) will also be enabled when log is set to a trace file.

Examples: Set Trace Command

When connected to the Communications Server, the following iimonitor commands turn on GCC and GCA tracing in that server to levels 3 and 5 respectively and write the output to trace file /tmp/gcc_log:

```
set trace level 3

set trace GCA 5

set trace log /tmp/gcc.log
```

The following iimonitor command closes the log file:

```
set trace log NONE
```

Register Server Command

The iimonitor register server command allows a GCF server to dynamically register its listen address with the Name Server.

Note: Using this command is the preferred method for restoring server registrations that have been lost or manually removed.

The register server command has the following format:

```
register server
```

Remove Tickets Command

The iimonitor remove tickets command removes installation password tickets stored by the Name Server.

Note: This command is valid for the Name Server only.

The remove tickets command has the following format:

```
remove [all]|local|remote tickets
```

all

Removes all tickets.

local

Removes local tickets only.

remote

Removes remote tickets only.

Remove Pooled Sessions Command

The iimonitor remove pooled sessions command terminates all DBMS sessions in the connection pool.

Note: This command is valid for the Data Access Server (GCD process) only.

This command has the following format:

```
remove pooled sessions
```

iinamu Command—Administer the Name Server

Permission required: Privileged user.

The Name Server Maintenance Utility (iinamu) allows a system administrator or other privileged user to display server information and administer the Name Server.

The iinamu command has the following format:

```
iinamu
```

At the IINAMU prompt, enter one of the following commands:

show [*svr_type*]

Shows the list of registered servers. The *svr_type* can be:

SERVERS

Shows all servers registered with the Name Server.

INGRES

(Default) Shows Ingres DBMS Server process

COMSVR

Shows GCC Communications Server process

IINMSVR

Shows Name Server process

DASVR

Shows Data Access Server process

STAR

Shows Star Server process

RMCMD

Shows Remote Command Server process

BRIDGE

Shows Protocol Bridge Server process

IUSVR

Shows Recovery Server process

gateway

Shows Enterprise Access Server (gateway) process. *Gateway* can be: MSSQL, ORACLE, DB2UDB, INFORMIX, SYBASE, RDB, or RMS.

user-defined DBMS

Shows user-defined DBMS server process, as defined in the server_class parameter for the DBMS Server in Configuration-By-Forms or Configuration Manager, for example BATCH, ONLINE, ACCOUNTS.

add *svr_type obj_name gcf_address* [*flag*]

Adds a server to the list of registered servers. Only a privileged user can run this command. Adding entries is not recommended and should be unnecessary as the information is automatically obtained from the servers during registration (at startup) and is based on the configured server_classes.

obj_name

Specifies the names of the objects that each server can service. Valid values are:

*** (asterisk)**—Is the default for DBMS Servers, which indicates that all databases can be accessed. All other servers do not register specific objects.

Database list—Is a single database or a comma separated list of databases that the Ingres DBMS Server can connect to.

flag can be one or more of the following values:

sole—Adds a sole server.

merge—Does not delete existing entries for the server at *gcf_address* when the new entry is added. Use this flag to add a new object to be serviced by an existing server.

delete *svr_type obj_name gcf_address*

Deletes a server from the list of registered servers. Only a privileged user can run this command. The *svr_type* and *obj_name* must associate with the information for the gcf_address.

stop

Stops the Name Server. This command is the correct way to stop the Name Server process. If the Name Server is stopped while servers are running, users cannot connect to those servers. Connected users will not be affected until they need to connect to a server. Only a privileged user can run this command.

help

Displays command information.

quit

Closes iinamu.

Note: At times, you may want to start DBMS servers that are not publicly registered with the Name Server. You can do this by setting the configuration parameter name_service to off. If this option is used, the server is not registered with the Name Server when it starts and is therefore invisible to iinamu. The GCF address can still be found and the server manually registered with the Name Server. If the server is a DBMS Server, connection can still be made by defining II_DBMS_SERVER to the server address.

iinamu Example: Show All Registered Servers

The following command shows all servers registered with the Name Server:

```
IINAMU> show servers
```

UNIX: Here is sample output:

```
IINMSVR     *     32770
IUSVR       *     32775
INGRES      *     32777
COMSVR      *     32779
COMSVR      *     32781
BRIDGE      *     32793
JDBC        *     32795
STAR        *     32799
RMCMD       *     32802
JOHN        *     32810
DASVR       *     32817
```

Windows: Here is sample output:

```
IINMSVR     *     EI\NMSVR\a10
IUSVR       *     EI\IUSVR\504
INGRES      *     EI\INGRES\f50
COMSVR      *     EI\COMSVR\e84
COMSVR      *     EI\COMSVR\cc8
BRIDGE      *     EI\BRIDGE\cb0
JDBC        *     EI\JDBC\ca8
STAR        *     EI\STAR\cd4
RMCMD       *     EI\RMCMD\cf8
JOHN        *     EI\JOHN\9b0
DASVR       *     EI\DASVR\924
```

The first column is the server type.

The second column is a list of databases registered to be served by the server. The entry * means that the server is registered to service requests for any database.

The third column is the server identifier, which is the GCF address for access to this server. This identifier can be used with the iimonitor command (in the case of DBMS, Recovery, and GCF servers).

iinamu Example: Show All DBMS Servers for the Server Class Ingres

This command shows all DBMS Servers:

```
IINAMU> show ingres
```

Here is sample output:

UNIX: Here is sample output:

```
INGRES * 3105
INGRES * 4204
```

Windows: Here is sample output:

```
INGRES * II\INGRES\aa
INGRES * II\INGRES\ca
```

The example shows two DBMS servers running at GCF addresses 3105 and 4204 (on UNIX), and with process IDs of aa and ca (on Windows).

iinamu Example: Show Communications Server Registrations

This command shows the Communications Server registrations:

```
IINAMU> show comsvr
```

UNIX: This sample output shows that two Communications Servers are running:

```
COMSVR * 3197
COMSVR * 3321
```

Windows: This sample output shows that two Communications Servers are running:

```
COMSVR * II\COMSVR\b3
COMSVR * II\COMSVR\a2
```

iinamu Example: Add a DBMS Server to the Name Server Registry

UNIX: This command adds to the Name Server registry a DBMS Server with GCF address 1093 that can connect to any database:

```
IINAMU> add ingres * 1093
```

This command adds a DBMS Server with GCF address 2180 that can connect only to the salesdb database:

```
IINAMU> add ingres salesdb 2180
```

This command adds the accounts database to the existing DBMS Server with GCF address 2180:

```
IINAMU> add ingres accounts 2180 merge
```

Windows: This command adds to the Name Server registry a DBMS Server with a process ID of "af" that can connect to any database:

```
IINAMU> add ingres * II\INGRES\af
```

This command adds a DBMS Server with a process ID of "ab" that can connect only to the salesdb database:

```
IINAMU> add ingres salesdb II\INGRES\ab
```

This command adds the accounts database to the existing DBMS Server II\INGRES\ab:

```
IINAMU> add ingres accounts II\INGRES\ab merge
```

iinamu Example: Delete a DBMS Server from the Name Server Registry

UNIX: This command deletes a DBMS Server with GCF address 1093 from the GCN registry so it is no longer visible from the Name Server:

```
IINAMU> delete ingres * 1093
```

Windows: This command deletes a DBMS Server with a process ID of "af" from the Name Server registry so it is no longer visible from the Name Server:

```
IINAMU> delete ingres * II\INGRES\af
```

iinamu Example: Stop the Name Server

This command stops the Name Server:

```
IINAMU> stop
```

iinethost Command—Echo Full Network Name of Local Host

The iinethost command echoes the fully qualified network host name of a given node in a network environment.

The iinethost command has the following format:

```
iinethost [outfile]
```

outfile

Echoes the host name to a specified output file.

iiodbcinst Command—Create ODBC Configuration File

Permission required: Access to the directory where the utility is located.

The iiodbcinst utility modifies or creates the ODBC configuration file, odbcinst.ini, on UNIX and VMS. For more information on configuring the Ingres ODBC driver, see the *Connectivity Guide*.

Note: The iiodbcinst utility does **not** write the ODBC data source configuration file odbc.ini.

The iiodbcinst command has the following format:

```
iiodbcinst [-batch] [-p altpath] [-rmpkg ] [-r]
```

-batch

Overwrites the odbcinst.ini file without prompting for confirmation.

-p *altpath*

Indicates an alternate target path for the odbcinst.ini file. The default value is /usr/local/etc or the definition of ODBCSYSINI.

-rmpkg

Removes the driver definition from the odbcinst.ini file.

-r

Forces all ODBC sessions to reject database updates (sets to read only).

iiodbcinst Example

The following example creates an ODBC configuration file in an alternate directory.

In the example, the driver manager is unixODBC, and the default search path of the unixODBC driver manager is /usr/local/etc. However, your installation does not allow non-privileged users to access /usr/local/etc.

To write an odbcinst.ini file to the path /ingres/odbcConfig, instead of the default /usr/local/etc, enter this command:

```
iiodbcinst -p /ingres/odbcConfig
```

When you invoke the Ingres ODBC Administrator (iiodbcadmin), the default system path is displayed as "ALTERNATE."

Since the default search path for unixODBC is different from your intended path, you need to define the environment variable ODBCSYSINI to execute ODBC applications. You may therefore have defined ODBCSYSINI prior to executing iiodbcinst. If this is the case, it is not necessary to supply the -p argument for iiodbcinst, because iiodbcinst already supports ODBCSYSINI. Furthermore, the Ingres ODBC Administrator displays the default search path as "SYSTEM."

See also iisuodbc Command (see page 125).

iipmhost Command—Echo Name of Host

The iipmhost command echoes the host name of a given node in a cluster environment.

The iipmhost command has the following format:

```
iipmhost [-node nodename] [-local] [outfile]
```

-node *nodename*

Echoes the host name of the corresponding node. The *nodename* can be either a node name or a node alias.

-local

Checks whether the value specified on the –node parameter is a node defined on the local host machine.

outfile

Echoes the host name to a specified output file.

iiremres Command—Remove Parameter from CONFIG.DAT

Permission required: Access to the directory where the utility is located.

The iiremres command removes a configuration parameter from config.dat and recalculates the value of any derived resources.

The iiremres command has the following format:

```
iiremres [-v] name
```

-v

Displays system commentary (verbose) to the standard output device as the operation continues.

name

Specifies the parameter name as it appears in the default configuration file.

iiremres Example

This command removes the parameter ii.lusilgpq0.gcn.local_node from the configuration file and recalculates any derived values:

```
iiremres -v ii.lusilgpqo.gcn.local_vnode
```

iisetres Command—Set Configuration Parameter

Permission required: Access to the directory where the utility is located.

The iisetres command sets a configuration resource in config.dat and recalculates derived resources.

The iisetres command has the following format:

```
iisetres [-v] [+p|-p] name [value]
```

-v

Displays system commentary to the standard output device as the operation continues.

+p

Protects the derived parameter from further automatic adjustments.

-p

Does not protect the derived parameter from further automatic adjustments.

name

Specifies the name of the configuration parameter as it appears in the config.dat file. For example: ii.*machinename*.config.date_alias.

value

Specifies the value of the configuration parameter. This parameter is optional if the +p or –p flag is set.

iisetres Examples

This command sets the configuration parameter default_page_size to 4096 in config.dat and recalculates derived resources:

```
iisetres ii.usilgpqo.dbms.*.default_page_size 4096
```

This command protects the derived parameter active_limit from further automatic derivation. Because no *value* is specified in the example, the value of the active_limit parameter will not be changed:

```
iisetres +p ii.usilgpqo.dbms.*.active_limit
```

iishowres Command—Display Memory Used by Locking and Logging

Valid on UNIX.

Permission required: Access to the directory where the utility is located.

The iishowres utility displays the amount of shared memory the locking and logging system uses to manage logging and concurrency in an installation. The size of the logging and locking memory segment depends on several parameters such as the number of concurrent users, the number of open databases, and the number of lock lists.

The iishowres command has the following format:

```
iishowres [-d] [-node nodename][-help]
```

-d

Shows the amount of memory each component of the locking and logging system uses.

If you do not use the -d flag, iishowres returns the total amount of shared memory needed by the logging and locking memory segment (in bytes).

-node *nodename*

Queries a specific node. A *nodename* is valid in a cluster installation only.

-help

Displays command syntax online.

iisunode Command—Set Up Node in a Cluster

Permission required: Installation owner.

The iisunode commands sets up additional nodes in a cluster. This utility is for use with Ingres Cluster Solution. After running iimkcluster utility on the first node in a cluster, run the iisunode utility on the remaining nodes in the cluster. See also iimkcluster Command (see page 102) and iiuncluster Command (see page 125).

The iisunode command has the following format:

```
iisunode [-remove]
```

-remove

Removes the node from the cluster.

iisuodbc Command—Run iiodbcinst Utility

Permission required: Access to the directory where the utility is located.

The iisuodbc command runs the iiodbcinst utility in interactive mode during the installation process. For details, see iiodbcinst Command (see page 120).

The iisuodbc command has the following format:

```
iisuodbc
```

iiuncluster Command—Convert Cluster to Standalone Instance

Permission required: Installation owner.

The iiuncluster utility converts a cluster Ingres instance to a non-cluster (stand-alone) Ingres instance. This utility is for use with Ingres Cluster Solution. See also iimkcluster Command (see page 102) and iisunode Command (see page 125).

The iiuncluster command has the following format:

```
iiuncluster
```

iivalres Command—Validate Configuration Resource

Permission required: Access to the directory where the utility is located.

The iivalres command validates a configuration resource for rule system constraint violations.

The iivalres command has the following format:

```
iivalres [-v] name value [rule_map]
```

-v

Displays system commentary to the standard output device as the operation continues

name

Specifies the name of the configuration parameter to set

value

Specifies the value of the configuration parameter

rule_map

Specifies the rule map file to use.

The rule map file contains a list of the rule system files (CRS extension files) to use when generating the default configuration.

iivalres Example

This command sees whether setting the default_page_size parameter to 2048 violates rule system constraints:

```
iivalres -v ii.usilgpqo.dbms.*.default_page_size 2048
```

iizic Command—Customize Time Zone Table Files

Permission required: Access to the directory where the utility is located.

The iizic utility customizes time zone table files so you can tailor time zone information for special cases not covered by the supplied time zone selections.

This utility works similarly to the UNIX zic utility.

The default *timezone_rule_file* is indicated by II_TIMEZONE_NAME variable. When II_TIMEZONE_NAME is set to the name of a time zone table, utilities such as date, and C functions such as localtime() and gettimeofday(), make GMT adjustments using the time zone table.

The Olsen time zone table for making Greenwich Mean Time (GMT) adjustments is supported. This method builds time zone tables based on the given rule file. The time zone tables consist of start and end GMT times for each time period. In addition, the corresponding GMT offset value and the abbreviation of the time zone name for each period are kept in the table.

When retrieving internal dates, which are stored in GMT, Ingres searches for the correct time period, and then applies the corresponding GMT offset value and the abbreviated time zone name to determine the local time.

The iizic command has the following format:

```
iizic [-doutput_directory] timezone_rule_file
```

-doutput_directory

Directs the output to the specified directory.

timezone_rule_file

Specifies the name of time zone rule file to customize.

iizck Command—Display Time Zone Table Files

Permission required: Access to the directory where the utility is located.

The iizck utility displays the time zone table files. You can check the time zone table currently in effect or a newly created one.

The display shows the Greenwich Mean Time (GMT) offset for the time zone. For those timelines adjusted for Daylight Savings Time (DST), it gives each date that the GMT offset changed.

If no parameters are specified, the iizck utility displays the II_TIMEZONE_NAME table.

The iizck command has the following format:

```
iizck [-name=timezone_name] [-fpathname/filename]
```

-name=*timezone_name*

Specifies the name of the time zone table currently in effect.

-f*pathname/filename*

Specifies the path and name of a new time zone table.

iizck Example

This command displays the time zone table currently in effect:

```
iizck
```

Here is sample output:

```
timezone name: australia-nsw
timezone file: c:\ingres::\ingres\files\zoneinfo\astrl\nsw

-----------------------------------------------------
       Period Begin                    GMT offset
       (YYYY_MM_DD HH:MM               (Minute)

       1971_10_31 02:00                660     EST
       1972_02_27 03:00                600     EST
       1972_10_29 02:00                660     EST
```

The dates range from 1971 to 2037.

imageapp Command—Build ABF or Vision Application Image

The imageapp command builds an application image from an Applications-By-Forms (ABF) or Vision application.

Note: This command replaces the abfimage command.

For more information about building application images, see the *Forms-based Application Development* Tools User *Guide* or *Character-based Querying and Reporting Tools User Guide*.

The imageapp command has the following format:

```
imageapp dbname |vnode::dbname[/server_class] applname [-oimagename] [-f]
[-constants_file='pathname'] [-5.0] [-w|+wopen] [-uusername] [-Ggroupid]
[-Rroleid]
```

dbname

Indicates the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

applname

Indicates the name of the application from which the image will be built.

-o*imagename*

Specifies the name of the new image.

-f

Forces recompiling of all objects in the application.

-constants_file='*pathname*'

Specifies the constant file to use.

-5.0

Specifies 5.0 compatibility mode.

-w

Suppresses display of warning messages.

+wopen

Directs ABF or Vision to display warnings when illegal OpenSQL statements are detected.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-Ggroupid

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

-Rroleid

Specifies a role identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-R*roleid*").

infodb Command—Display Database Information

The infodb command displays a variety of information on a database, including its status, the location of its files (from the configuration file aaaaaaaa.cnf), and a history of checkpoints and journaling.

To use this command, you must be a privileged user or the DBA of the specified database. If you are a privileged user, you can use the -u flag to impersonate another user. On VMS, to use this command against a database in a group level installation, you must be a privileged user (VMS CMKRNL, SYSPRV, and PHY_IO privileges).

The infodb command has the following format:

```
infodb [dbname[/server_class]][#c[n]][-uusername][-help]
```

dbname

Indicates the name of the database, and if required, the *server_class*, as described in Standard Flags and Parameters (see page 13).

If no database is specified, infodb prints a report for each database.

#c[*n*]

Provides detailed information about a specific checkpoint for the database. The checkpoint number *n* must be a valid checkpoint number. If *n* is omitted, information about the most recent completed checkpoint is displayed.

UNIX: In bash shell, you must place this option in quotes; otherwise characters after the # will be treated as a comment. For example:

```
infodb empdata "#c1"
```

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-help

Displays command syntax online.

Infodb Command Output – Database Information Section

The following is sample output from the database information section. This guide includes the callouts (1) and (2) to aid in explanation:

```
(1)================11-FEB-2008 14:05:55.02 Database Information==========

    Database : (doc,kbref)  ID : 0x2D5BF682
    Extents  : 5    Last Table Id : 171
    Config File Version Id : 0x00040001   Database Version Id : 5
    Status   : VALID,JOURNAL,CKP,DUMP,ROLL_FORWARD,CFG_BACKUP

(2)             The Database has been Checkpointed.
                The Database is Journaled.

                Journals are valid from checkpoint sequence : 1
```

Fields are as follows:

At (1)

Identifies the date and time the infodb operation was run.

Database

Identifies the name (doc) and owner (kbref) of the database.

ID

Shows the internal identifier of the database.

Extents

Indicates the number of locations the database is using.

Last Table ID

Indicates the integer identifier assigned to the last created table.

Config File Version ID

Shows the major (upper 2 bytes) and minor (lower 2 bytes) versions of the configuration file.

Database Version ID

The version of DMF that created the database. Note that this is *not* related to the Ingres version of the database.

Status

Displays status information for the database. Status abbreviations are as follows:

CFG_BACKUP—automatic backup of the configuration file is enabled.

CKP—indicates that you must perform a rollforward +c (back to saveset) *before* you can do a rollforward -c +j..

DUMP—the database has undergone dump processing (that is, a dump file was created in the dump location) via some online checkpoint

JOURNAL—the database is journaled.

JOURNAL_DISABLED—journaling has been disabled.

NOLOGGING—the database has been opened by a set nologging session. Note that if this session encounters an error, the database will be marked inconsistent.

ROLL_FORWARD—indicates that rollforward is available on the database and has not been run to completion since the last checkpoint was taken.

SMINC—indicates the system catalogs are in an inconsistent state.

VALID—the database is consistent and available for use. If this does not appear, the database is marked inconsistent.

At (2)

This section displays comments on the status of the database. Important state information is shown.

The Database is Inconsistent. Cause of Inconsistency: <...>

Shown if the database is inconsistent. The cause of inconsistency can be one of the following:

NOLOGGING_ERROR—a transaction failed while the database was in the nologging state.

NOLOGGING_OPENDB—the database was opened for the first time, but was in the nologging state. This means a session exited abnormally.

OPEN_COUNT—the database was opened for the first time, but the database open count in the configuration file was *not* zero. This means the configuration file could not be read during a recovery attempt.

REC_OPEN_FAILURE—the RCP could not recover a database because the database could not be opened.

RECOVER_ERROR—the RCP failed to recover a database due to an unexpected logging system or recovery protocol problem.

REDO_ERROR—the RCP failed to recover a database due to an error in REDO processing.

RFP_FAIL—the rollforward of the database level checkpoint failed.

UNDO_ERROR—the RCP failed to recover a database due to an error in UNDO processing.

WILL_COMMIT_ERR—the RCP was unable to restore a transaction to the willing commit state.

The Database has been Checkpointed.

Shown if the database has been checkpointed.

The Database is Journaled. or The Database is not Journaled.

Shows the journaling status.

Journaling has been disabled on this database by alterdb. Run 'ckpdb +j' to re-enable journaling.

Shown if journaling has been disabled.

Database is being accessed with Set Nologging, allowing transactions to run while bypassing the logging system.

Shown if a set nologging session is active on the database.

Journals are valid from checkpoint sequence: *checkpoint sequence number*

Shows the earliest checkpoint from which rollforward is allowed.

Journals are not valid from any checkpoint.

Shown if rollforward is not valid from any checkpoint, or there are no checkpoints.

Infodb Command Output – Journal Information Section

The following is sample output from the journal information section:

```
----Journal information-------------------------------------------------
    Checkpoint sequence :      3    Journal sequence :               3
    Current journal block :    2    Journal block size :         16384
    Initial journal size :     4    Target journal size :          512
    Last Log Address Journaled : <760829464:19666:188>
```

Fields are as follows:

Checkpoint sequence

Indicates the current checkpoint sequence number. Incremented when a checkpoint operation is performed.

Journal sequence

Indicates the current journal file sequence number.

Current journal block

Indicates the current journal file block sequence number. This is the logical end-of-file of the current journal file.

Journal block size

Indicates the block size of the current journal file, in bytes.

Initial journal size

Indicates the number of blocks initialized in the "first journal file" when it is created. The first journal file is the journal file created during the checkpoint +j operation. Subsequent journal files created before the next checkpoint is done will *not* be initialized.

Target journal size

Indicates the size, in blocks, to which the current journal file may grow before a new journal file should be created. A new journal file will be created at the start of the next archive cycle after the current journal file reaches this size.

Last log address journaled

Indicates the log address (*log sequence number*, *log page number*, *log word offset*) of the last log record written to a journal file.

Infodb Command Output – Dump Information Section

The following is sample output from the dump information section:

```
----Dump information----------------------------------------------------
    Checkpoint sequence :      3    Dump sequence :                1
    Current dump block :       1    Dump block size :          16384
    Initial dump size :        4    Target dump size :           512
    Last Log Address Dumped : <760829464::21100:100>
```

Fields are as follows:

Checkpoint sequence

Indicates the current checkpoint sequence number. Incremented when a checkpoint operation is performed.

Dump sequence

Indicates the current dump file sequence number.

Current dump block

Indicates the current dump file block sequence number. This is the logical end-of-file of the current dump file.

Dump block size

Indicates the block size of the current dump file, in bytes.

Initial dump size

Indicates the initial allocation of the current dump file, in blocks. The number of blocks initialized when a dump file is created.

Target dump size

Indicates the size, in blocks, to which the current dump file may grow before a new dump file should be created. A new dump file will be created at the start of the next archive cycle after the current dump file reaches this size.

Last log address dumped

Indicates the log address (*log sequence number*, *log page number*, *log word offset*) of the last log record written to a dump file.

Infodb Command Output – Checkpoint History for Journal Section

The following is sample output from the checkpoint history for journal section:

```
----Checkpoint History for Journal-------------------------------------
    Date                Ckp_sequence  First_jnl  Last_jnl valid  mode
    ------------------------------------------------------------------
    11-NOV-2008 13:23:43.57       1       1          1        1  OFFLINE
    11-NOV-2008 13:24:50.40       2       2          2        1  ONLINE
    11-NOV-2008 13:58:52.65       3       3          3        1  ONLINE,
                                                                  TABLE
```

Fields are as follows:

Date

Indicates the date and time the checkpoint operation was done.

Ckp_sequence

Indicates the sequence number of the checkpoint.

First_jnl

Indicates the journal sequence number of the first (or oldest) journal file corresponding to the checkpoint.

Last_jnl

Indicates the journal sequence number of the last (or youngest) journal file corresponding to the checkpoint.

Valid

Indicates whether the checkpoint is valid (1 implies valid, 0 implies invalid).

Mode

Indicates whether the checkpoint operation was online or offline. Also indicates TABLE if the checkpoint was a table checkpoint.

To recover the entire database, you must specify #c2, for example, in the rollforwarddb command to roll forward from the database checkpoint. Checkpoint 3 was taken on selected tables.

Infodb Command Output – Checkpoint History for Dump Section

The following is sample output from the checkpoint history for dump section:

```
----Checkpoint History for Dump----------------------------------------
    Date                Ckp_sequence  First_dmp  Last_dmp valid  mode
    ------------------------------------------------------------------
    11-NOV-2008 13:58:52.65    2       1          1         1    ONLINE
    11-NOV-2008 13:58:52.65    3       1          1         1    ONLINE,
                                                                 TABLE
```

Fields are as follows:

Date

Indicates the date and time the checkpoint operation was done.

Ckp_sequence

Indicates the sequence number of the checkpoint.

First_dmp

Indicates the dump sequence number of the first (or oldest) dump file corresponding to the checkpoint.

Last_dmp

Indicates the dump sequence number of the last (or youngest) dump file corresponding to the checkpoint.

Valid

Indicates whether the checkpoint is valid (1 implies valid, 0 implies invalid).

Mode

Indicates whether the checkpoint operation was online or offline (should always be online). Also indicates TABLE if the checkpoint was a table checkpoint.

Infodb Command Output – Cluster Journal History Section

The following is sample output from the cluster journal history section:

```
----Cluster Journal History----------------------------------------
    Node ID   Current Journal   Current Block   Last Log Address
    ------------------------------------------------------------
          0                1               1   <760829462:4731:1592>
          1                3              10   <760829464:35000:188>
```

Fields are as follows:

Node ID

Indicates the integer identifier of the node.

Current journal

Indicates the node's current journal file sequence number.

Current block

Indicates the node's current journal file block sequence number. This is the logical end-of-file of the node's current journal file.

Last log address

Indicates the log address (*log sequence number*, *log page number*, *log word offset*) of the last log record written to a journal file on this node.

Infodb Command Output – Extent Directory Section

The following is sample output from the extent directory section, which shows all locations used by the database:

```
----Extent directory-------------------------------------------------------------

    Location       Flags        Physical_path
    ------------------------------------------------------------------------------
    ii_database    ROOT,DATA    c:\Ingres\IngresII\ingres\data\default\iidbdb
    ii_journal     JOURNAL      c:\Ingres\IngresII\ingres\jnl\default\iidbdb
    ii_checkpoint  CHECKPOINT   c:\Ingres\IngresII\ingres\ckp\default\iidbdb
    ii_dump        DUMP         c:\Ingres\IngresII\ingres\dmp\default\iidbdb
    ii_work        WORK         c:\Ingres\IngresII\ingres\work\default\iidbdb
```

Fields are as follows:

Location

Identifies the logical name of the location.

Flags

Indicates the types of database files stored in the location. The possibilities are:

ALIAS—this is a location alias. This means at least one other location points to the same area as this location. This flag is used only for checkpoint and rollforward operations so that locations are neither checkpointed nor rolled forward more than once.

AWORK—this is an auxiliary work file location. Work files (that is, for sorts and temporary tables) are stored in the location.

CHECKPOINT—checkpoint files are stored in the location.

DATA—user data (such as tables, indexes) is stored in the location.

DUMP—dump files are stored in the location.

JOURNAL—journal files are stored in the location.

ROOT—system data (that is, system catalogs) is stored in the location.

WORK—work files (for sorts and temporary tables) are stored in the location.

Physical_path

Identifies the physical path of the location.

ingmenu Command—Start Ingres Menu

The ingmenu command invokes Ingres Menu, a forms-based interface for accessing the Ingres tools. For a complete description of Ingres Menu, see the *Character-based Querying* and *Reporting Tools User Guide*.

The ingmenu command has the following format:

```
ingmenu dbname|vnode::dbname[/server_class] [-e] [-uusername] [-Ggroupid]
```

dbname

Specifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-e

Invokes Ingres Menu in empty mode. This flag is passed to the QBF, RBF, TABLES, and VIFRED options of Ingres Menu. It causes a catalog of applications, Join Definitions, tables, reports, or other objects to be displayed empty initially, so that you can enter specific names for those objects.

-uusername

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-Ggroupid

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

ingmenu Examples

1. Invoke the Ingres Menu on the employee database:

   ```
   ingmenu employee
   ```

2. Invoke the Ingres Menu in expert mode with empty catalogs, on the projects database on the hq node:

   ```
   ingmenu hq::employee -e
   ```

ingnet Command—View and Define Ingres Net Node Definitions

Permission required: Access to the directory where the utility is located.

The Network Utility (ingnet) is a stand-alone tool that permits you to view and define Ingres Net node definitions, which then allows you to connect to remote Ingres installations through Ingres Net. The ingnet utility also allows you to launch the stand-alone Database Object Manager, Monitor, and SQL Scratchpad windows for such installations.

The ingnet command has the following format:

```
ingnet [-vnode=vnode]
```

-vnode=*vnode*

Specifiesthe name of the remote node on which the connection information is to be stored. This vnode must have been configured previously through either the ingnet or netutil utilities.

ingstart Command—Start an Ingres Instance

Permission required: Installation owner, privileged user.

The ingstart command starts an Ingres installation. It also checks that you have sufficient operating system resources to run Ingres and have initialized the log file.

The ingstart utility starts all the elements of your installation in the correct sequence. It starts the Name Server, recovery and archiver processes, and DBMS servers. If you are authorized to run them, it also starts the Communications Server, Visual DBA Remote Command Server, and Star Server.

Note: If a server or process is specified, only that component is started.

The ingstart command has the following format:

```
ingstart [-iigcn | -dmfrcp | -dmfacp | -client | -rmcmd | -icesvr |
[-iidbms|-iigcc|-iigcb|-iigcd|-iistar|-oracle|-informix|-mssql|
 -sybase|-db2udb|-rdb|-rms [= config_name]]] [-cluster]
[-node nodename] [-help]
```

-iigcn

Starts the Name Server.

-dmfrcp

Starts the recovery process.

-dmfacp

Starts the archiver process.

-client

Starts a client service by starting only a Name Server and Communications Server. Valid on Windows only.

-rmcmd

Starts the Remote Command Server required by Visual DBA.

-iidbms

Starts the DBMS Server. You can optionally specific a *config_name* as the name of the server.

-iigcc

Starts the Communications Server. You can optionally specify a *config_name* as the name of the server.

-iigcb

Starts the Bridge Server. You can optionally specify a *config_name* as the name of the server.

-iigcd

Starts the Data Access Server. You can optionally specify a *config_name* as the name of the server.

-iistar

Starts the Star Server. You can optionally specify a *config_name* as the name of the server.

-informix

Starts Enterprise Access for Informix. Valid on UNIX and Windows only.

-mssql

Starts Enterprise Access for MS SQL Server. Valid on Windows only.

-oracle

Starts Enterprise Access for Oracle.

-rdb

Starts Enterprise Access for RDB. Valid on VMS only.

-rms

Starts Ingres RMS Access. Valid on VMS only.

-sybase

Starts Enterprise Access for Sybase. Valid on UNIX and Windows only.

-db2udb

Starts Enterprise Access for IBM DB2 UDB. Valid on UNIX and Windows only.

config_name

Specifies the name of the server being started. To see a list of server names, click the Configure tab in Configuration Manager.

-cluster

Starts Ingres on all nodes in the cluster. Valid in a cluster installation only.

-node *nodename*

Starts Ingres on the specific node. Valid in a cluster installation only.

-help

Displays command syntax online.

ingstart Examples

This command starts an additional default DBMS Server:

```
ingstart -iidbms
```

This command starts the "speedy" Communications Server:

```
ingstart -iigcc=speedy
```

On UNIX, this command starts the installation interactively using the configuration option, where $II_SYSTEM is set to /install/r6:

```
setenv II_SYSTEM /install/r6
ingstart
```

On UNIX, this command starts the installation automatically by including ingstart in the /etc/rc or other boot script, where *userid* is the user ID defined during installation:

```
su userid -c /install/r6/ingres/utility/ingstart \
   > /dev/console
```

ingstop Command—Stop an Ingres Instance

Permission required: Installation owner, privileged user.

Important! To use this command, you must be logged into the installation owner account.

The ingstop command shuts down an Ingres installation. It stops the servers in an orderly fashion for reconfiguration or system shutdown. It automatically brings down all or selected server-related processes in the installation. It can shut down servers, the archiver and recovery processes, and deallocate shared memory.

The ingstop command provides a graceful shutdown: the program waits for all traffic to terminate and for all users to exit from Ingres before shutting down the Ingres processes. You can optionally specify a forced shutdown.

Because it is important that processes be brought down in the correct sequence, you should use ingstop whenever you shut down the entire installation. You can also use ingstop to shut down the locking and logging system.

Note: On VMS, the ingstop command has no flags or parameters.

The ingstop command has the following format:

```
ingstop [-iigcn|-dmfrcp|-dmfacp|-client|-rmcmd|-icesvr|
[-iidbms|-iigcc|-iigcb|-iigcd|-iistar|-oracle|-informix|-mssql|
 -sybase|-db2udb|-rdb|-rms [= connect_id]]]
[-f] [-timeout=minutes] [-kill] [-show |-check] [-force|-immediate]
[-cluster] [-node nodename] [-help]
```

-iigcn

Stops the Name Server.

-dmfrcp

Stops the recovery process.

-dmfacp

Stops the archiver process.

-client

Stops a client service by stopping only a Name Server and Communications Server. Valid on Windows only.

-rmcmd

Stops the Remote Command Server required by Visual DBA.

-iidbms

Stops the DBMS Server.

-iigcc

Stops the Communications Server.

-iigcb

Stops the Bridge Server.

-iigcd

Stops the Data Access Server.

-iistar

Stops the Star Server.

-informix

Stops Enterprise Access for Informix. Valid on UNIX and Windows only.

-mssql

Stops Enterprise Access for MS SQL Server. Valid on Windows only.

-oracle

Stops Enterprise Access for Oracle.

-rdb

Stops Enterprise Access for RDB. Valid on VMS only.

-rms

Stops Ingres RMS Access. Valid on VMS only.

-sybase

Stops Enterprise Access for Sybase. Valid on UNIX and Windows only.

-db2udb

Stops Enterprise Access for IBM DB2 UDB. Valid on UNIX and Windows only.

connect_id

Specifies the connect ID of the server to be stopped. To see a list of DBMS server connect IDs, use the iinamu command.

-f

Forces immediate shutdown.

-timeout=*minutes*

Waits the specified number of minutes for active sessions to terminate before shutting down the installation.

-kill

Shuts down the installation without waiting for currently executing transactions to complete. Transaction recovery will be required when the installation is restarted. Any Ingres processes that cannot be shut down by conventional means are terminated.

-force

Forces the shut down of active servers in the installation without waiting for users to disconnect.

-immediate

Shuts down the installation immediately. It does not wait for currently executing transactions to complete. Transaction recovery will be required when the installation is restarted.

-show | -check

Displays a list of currently running Ingres processes, but does not shut them down.

-cluster

Shuts down Ingres on all nodes in the cluster. Valid in a cluster installation only.

-node *nodename*

Shuts down Ingres on the specified node. Valid in a cluster installation only.

-help

Displays command syntax online.

ingprenv Command—Display Environment Variable Value

The ingprenv command displays values for the Ingres environment variables defined at installation time. This command reads the symbol table (symbol.tbl). For more information on the symbol.tbl file, see the *System Administrator Guide*.

Note: This command is not available on VMS. Use the VMS command SHOW LOGICAL instead.

The ingprenv command has the following format:

```
ingprenv [variable_name]
```

variable_name

Specifies the name of the environment variable. If no name is specified, all variables are displayed. If the variable is not defined, no output is displayed.

ingprenv Example

This command displays the value of the II_DATABASE environment variable:

```
ingprenv II_DATABASE
```

ingsetenv Command—Set Ingres Environment Variable

The ingsetenv command sets or changes the value of an Ingres environment variable. Ingres environment variables are stored in the Ingres symbol table. To view variable settings, use the ingprenv command.

This command affects Ingres environment variables stored in the symbol table (symbol.tbl) only. It does not affect Windows or UNIX environment variables. For more information on the symbol table, see the *System Administrator Guide*.

Caution: *Never edit the symbol table directory.*

Note: The ingsetenv command is not available on VMS. Use the VMS command DEFINE instead.

The ingsetenv command has the following format:

```
ingsetenv variable_name value
```

variable_name

Specifies the environment variable that you want to set or change.

value

Specifies the value to which you want to set the variable.

ingsetenv Example

This command sets the Applications-By-Forms directory environment variable (ING_ABFDIR):

Windows:

```
ingsetenv ING_ABFDIR \proj\abf
```

UNIX:

```
ingsetenv ING_ABFDIR /proj/abf
```

ingunset Command—Delete Environment Variable

The ingunset command deletes or unsets the specified Ingres environment variable from the Ingres symbol table. To view variable settings, use the ingprenv command.

This command affects only Ingres environment variables stored in the symbol table (symbol.tbl), not Windows or UNIX environment variables. For more information on the symbol.tbl file, see the *System Administrator* Guide.

Caution: *Never edit the symbol table directory.*

Note: The ingunset command is not available for VMS. Use the VMS command DEASSIGN instead.

The ingunset command has the following format:

```
ingunset variable_name
```

variable_name

Specifies the installation variable that you want to unset.

ipm Command—Start the Interactive Performance Monitor

The ipm command invokes the forms-based Interactive Performance Monitor, which combines the functions of the lockstat, logstat, iimonitor, and iinamu utilities in a single tool.

The Interactive Performance Monitor is used to view various aspects of a running installation. It can be used to view a running server, examine the logging and locking system, and perform actions on active servers.

For complete details on the use of the IPM command, see the *Interactive Performance Monitor User Guide*.

The ipm command has the following format:

```
ipm [-ddbname] [-e] [i] [-l] [-lrestype] [-n] [-t] [-rseconds] [-s] [-help]
```

-d*dbname*

Reports only on resources for database *dbname*.

-e

Displays system and user locklists.

-i

Displays interactive sessions and user locklists.

-l

Report on all resource types.

-l*restype*

Reports on specified resource type (page, table, database, and so on) only. If *restype* is not specified, reports on all resource types.

-n

Does not print resources granted in null mode.

-t

Reports on a particular table. With this option, you must also specify the -d*dbname* parameter.

-r*seconds*

Sets refresh time for various screens.

-s

Runs ipm in standalone mode. Databases are not opened.

-help

Displays command syntax online.

iquel Command—Start Interactive QUEL Terminal Monitor

The iquel command invokes the forms-based Terminal Monitor for interactive QUEL (IQUEL). For a complete description of the forms-based Terminal Monitor, see the *Character-based Querying and Reporting Tools User Guide*.

The iquel command has the following format:

```
iquel [SQL option flags] dbname|vnode::dbname[/server_class]
```

SQL option flags

Specify options passed to the Terminal Monitor when invoked. These include the standard flags (-u*username*, -G*groupid*, and -R*roleid*), and formatting and DBMS control flags. For details, see sql (see page 245).

dbname

Identifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

iquel Example

This command invokes interactive forms-based QUEL on the employee database:

```
iquel employee
```

isql Command—Start Interactive SQL Terminal Monitor

The isql command invokes the forms-based Terminal Monitor for interactive SQL (ISQL). For a complete description of the forms-based Terminal Monitor, see the *Character*-based *Querying and Reporting Tools User Guide*.

The isql command has the following format:

```
isql [SQL option flags] dbname|vnode::dbname[/server_class]
```

SQL option flags

Specify options passed to the Terminal Monitor when invoked. These include the standard flags (**-u***username*, -G*groupid*, and -R*roleid*), and formatting and DBMS control flags. For details, see sql (see page 245).

dbname

Identifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

isql Examples

This command invokes interactive forms-based SQL on the employee database on a local node:

```
isql employee
```

This command invokes interactive forms-based SQL on the employee database on the hq remote node:

```
isql hq::employee
```

This command opens empdata, displays float4 columns in G format with two decimal places and integer1 columns with three spaces:

```
isql -f4g12.2 -i13 empdata
```

ivm Command—Start Ingres Visual Manager

Permission required: Access to the directory where the utility is located.

The ivm command starts Ingres Visual Manager (IVM).

The Ingres Visual Manager is a graphical user interface that provides a global view into the Ingres installation. It is a system console from which you can manage Ingres components and access other utilities.

You can use IVM to start and stop the Ingres, monitor the status of the installation or individual servers, view and configure system, user, and other types of parameters, view log files and event statistics for the installation or individual servers, and view and define error message alerts.

This utility captures events that are occurring in the system and allows them to be filtered for emphasis.

The ivm command has the following format:

```
ivm
```

lartool Command—Start Logging, Archiving, and Recovery Utility

Permission required: System administrator. On VMS, OpenVMS privileges.

The lartool command starts the logging, archiving, and recovery utility.

The lartool (logging, archiving, and recovery) utility permits the system administrator to manually abort or commit distributed transactions without bringing down the entire installation. A distributed transaction may need to be manually aborted or committed if it is left in a willing commit state following the loss of the coordinator of the distributed transaction.

This utility can also abort orphaned transactions that cannot be removed by removing the session owner.

The lartool command has the following format:

```
lartool [-help]
```

-help

Displays the lartool utility commands online.

lartool Example

This example shows how to use the lartool utility to abort or commit a transaction:

1. Identify the ID of the transaction you wish to abort or commit, by using ipm or logstat.
2. Enter the command **lartool**. The utility prompts you to select an operation, either ABORT or COMMIT.
3. Select the operation. You are prompted for the transaction ID (TX_ID). Lartool then performs the requested operation.
4. To exit from lartool, type the command: **exit**.

Caution! *Use this utility for a well-defined purpose only. Be aware that if you are aborting a transaction, it may take time to complete. In this case, do not bring the server down. Doing so merely transfers the rollback chore to the recovery process instead of the DBMS Server process. The rollback must still be processed, and the transaction will not be fully removed until rollback processing is completed.*

lockstat Command—Display Locking Status

Permission required: Installation owner.

The lockstat utility displays locking status information for your Ingres installation. It allows you to examine the state of the Lock Database by providing a summary listing and a snapshot of the installation's locking activity.

This tool is useful for finding lock contention and concurrency problems. It will help you identify locking bottlenecks so that you can correct the problem by setting lockmode appropriately or by remodeling the application if it is a fundamental design or program flow problem.

Note: The forms-based ipm utility incorporates the lockstat functions.

The lockstat command has the following format:

```
lockstat [-summary | -statistics | -lists | -user_lists |
-special_lists | -resources | -dlm | -dlmall | -dirty | -help]
```

lockstat

Displays all reports. This is the default.

-summary

Displays locking system quotas

-statistics

Displays locking system quotas and locking system summary

-lists

Displays locks by lock list (user and special)

-user_lists

Displays locks by lock list (user)

-special_lists

Displays locks by lock list (special, that is, "NONPROTECT")

-resources

Displays locks by resource

-dlm

Displays the cluster DLM (Distributed Lock Manager) view of locks (grouped by resource) that have one or more waiters. The -dlm and -dlmall options are available only in releases that support clusters and when Ingres is configured for such support.

Note: When running in an installation configured with cluster support, lockstat output is the same as the non-cluster lockstat but with the -dlm output appended to the end.

-dlmall

Displays the same information as lockstat -dlm, but for all locks in the installation (that is, including those that do not have locks in a wait or convert status)

-dirty

Displays the requested information without waiting for mutexes. Using the -dirty option means the information displayed may be out of date or inconsistent.

-help

Displays command options online

Lockstat Command Output

The key items to examine in lockstat output are the waiting statistics, including the following:

- Lock wait in the Locking System Summary
- Deadlock in the Locking System Summary
- Wait in the Locks by Lock List
- Status WAIT in the Locks by Lock List

Note: The lockstat command gives detailed statistics on *all* locking activity in the installation, so if there is much activity, the quantity of output will be considerable.

Lockstat Command Output – Locking System Quotas

The Locking System Quotas portion of the sample output is a summary listing of locking quotas for the installation. All values are cumulative from the time ingstart was run for this iteration of the system.

Here is sample output from this section:

```
===========Fri Apr 25 13:34:06    Locking System Quotas=========
  Total Locks          65000  Total Resources     65000
  Locks per transaction                             500
  Lock hash table      12983  Locks in use          129
  Resource hash table  12983  Resources in use      127
  Total lock lists      1480  Lock lists in use      57
```

Fields are as follows:

Total locks

Maximum number of locks in the installation

Total Resources

Maximum number of lockable resources in the installation

Locks per transaction

Maximum number of locks that may be acquired by a transaction

Lock hash table

Number of hash buckets in the locking system hash table

Locks in use

Number of locks currently in use in the installation

Resource hash table

Number of hash buckets in the resource hash table

Resources in use

Total number of countable resources in use

Total lock lists

Maximum number of lock lists available

Lock lists in use

Number of lock lists currently in use

Lockstat Command Output – Locking System Summary

The Locking System Summary portion of the sample output is a summary listing of locking activity for the installation. All values are cumulative from the time ingstart was run for this iteration of the system.

Here is sample output from this section:

```
===========Fri Nov 13 13:34:06    Locking System Summary========
  Create lock list   49395    Release lock list    49327
  Request lock      586295    Re-request lock     457810
  Convert lock       68157    Release lock        410846
  Escalate              10    Lock wait            27489
  Convert wait           2    Convert Deadlock         1
  Deadlock Wakeups    2218    Max dlk queue len       10
  Deadlock Search     1947    Deadlock               134
  Cancel               135    Convert Search           2
  Allocate CB      1161478    Deallocate CB      1160861

  LBK Highwater       3    LLB Highwater         87
  SBK Highwater       5    LKB Highwater        428
  RBK Highwater       5    RSB Highwater        420
  Max Local dlk srch  3    Dlk locks examined  2361
  Max rsrc chain len  5    Max lock chain len     5
  Callback Wakeups    0    Callbacks Invoked      0
  Callbacks Ignored   0
```

Fields are as follows:

Create lock list

Number of times a lock list was created for server, session, or transaction

Release lock list

Number of times a release of a lock list occurred for a server, session, or transaction

Request lock

Number of new lock requests that the locking system processed

Re-request lock

Number of times an implicit lock conversion request was issued on a resource that the lock list already had locked. Implicit lock conversion requests can occur when a request is made on a page for update that was previously requested for read.

Convert lock

Number of times an explicit lock conversion request is made to change a lock mode on a physical lock from one mode to another. These types of requests occur as a result of a physical lock being converted during an existing transaction to lower or higher modes.

Release lock

Number of times a specific logical lock is released, as opposed to a full, partial, or physical lock release.

Escalate

Number of times a partial release occurred to allow lock escalation from page to table level

Lock wait

Number of times a new lock request had to wait to be granted

Convert wait

Number of times an existing lock waited for conversion to a different lock mode

Convert deadlock

Number of times a request for conversion turned into a deadlock

Deadlock Wakeups

Number of times the interval-based deadlock detection thread was awakened

Max dlk queue len

Maximum number of waiting lock lists examined by the deadlock detection thread

Deadlock search

Number of times a deadlock search was initiated

Deadlock

Number of times that deadlock existed

Cancel

Number of times a lock request was canceled due to a time-out or interrupt

Convert search

Number of times a convert deadlock search was initiated. The searches are performed when converting a lock from one mode to another.

Allocate CB

Number of locking control block allocations

Deallocate CB

Number of locking control block deallocations

LBK Highwater

Maximum number of lock list blocks allocated

LLB Highwater

Maximum number of lock lists allocated

SBK Highwater

Maximum number of lock blocks allocated

LKB Highwater

Maximum number of locks allocated

RBK Highwater

Maximum number of resource blocks allocated

RSB Highwater

Maximum number of resources allocated

Max Local dlk srch

Maximum number of locks examined to resolve

Dlk locks examined

Number of locks examined by the deadlock detection thread

Max rsrc chain len

Maximum length of a resource hash chain

Max lock chain len

Maximum length of a lock hash chain

The remaining fields are relevant only when the installation has been configured to run with the Distributed Multi-Cache Management (DMCM) protocol:

Callback Wakeups

Number of times the DMCM callback thread was awakened

Callbacks Invoked

Number of times callback functions were invoked to resolve a blocking cache lock

Callbacks Ignored

Number of blocking cache locks which had already been released by the time the callback function was invoked

Lockstat Command Output – Locks by Lock List

The "Locks by lock list" portion of the lockstat utility prints the lock information sorted by lock list. The first line item reports the lock list identifier. Any locks associated with the specified lock list are listed following the lock list description and indented.

Any locks associated with the particular lock list are listed following the lock list description and indented.

Most lock lists represent transaction units and hold the locks owned by their transactions. Some lock lists are used to hold special server or cache locks required for processing; these lock lists are owned and managed by the DBMS Server or recovery process rather than by user transactions.

Here is sample output from this section:

```
--------------------------Locks by lock list---------------------
Id: 00000001 Tran_id: 0000000000000001A R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/250) Status: NONPROTECT,NOINTERRUPT PID: 10781
SID:0000000E
Id: 00000002 Tran_id: 0000000000000019 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/250) Status: NONPROTECT,NOINTERRUPT PID: 10781
SID:00000009
Id: 00000003 Tran_id: 0000000000000018 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:0000000C
Id: 00000004 Tran_id: 0000334C3362D62B R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/250) Status: NONPROTECT PID:
10781 SID:00000007
Id: 00000005 Tran_id: 0000000000000016 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:00000007
Id: 00000006 Tran_id: 0000000000000015 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:0000000B
Id: 00000007 Tran_id: 0000000000000014 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:00000005
Id: 00000008 Tran_id: 0000000000000013 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/250) Status: NONPROTECT,NOINTERRUPT PID: 10781
SID:00000006
Id: 00000009 Tran_id: 0000000000000012 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (1,0/250) Status: NONPROTECT,NOINTERRUPT PID:
10781 SID:00000001
  Id: 00000026 Rsb: 0000003C Gr: N  Req: N  State: GR PHYS(1)
KEY(AUDIT,LABEL_CACHE)
Id: 0000000A Tran_id: 0000000000000011 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (0,0/0) Status: NONPROTECT,PSHARED PID:10781
SID:00000001

Id: 0000000B Tran_id: 0000000000000010 R_llb: 00000000 R_cnt: 0
Wait: 00000000 Locks: (95,0/250) Status: NONPROTECT,NOINTERRUPT,
SHARED PID:10781 SID:00000001

Id: 0000001F Rsb: 00000034 Gr: IS Req: IS State: GR PHYS(1)
```

Fields are as follows:

Id

Internal lock list identifier (lock list block)

Tran_id

Transaction identifier associated with this lock list. This value correlates to a transaction identifier in the logstat utility output.

R_llb

Related lock list identifier, if not a transaction lock list

R_cnt

Number of related lock list identifiers that this lock list must assure are released before this lock list can be released

Wait

Internal resource block identifier of the lock that is currently blocked

Locks

Made up of three values: total number of locks currently on the list, number of logical locks on the list currently, and total number of locks allowed to be on this list

STATUS

Indicates the current state of the lock list. The possible values are:

WAIT—waiting for lock

NONPROTECT—can be released without going through recovery (system lock lists)

ORPHAN—lock list remaining without transaction

EWAIT—waiting for system event

RECOVER—lock list taken over by the recovery process

MASTER—lock list owned by the recovery process

ESET—lock list set on wait queue for event

EDONE—event that lock list is waiting for is done

NOINTERRUPT—lock requests on this list are non-interruptible

PID

Process ID of the lock list owner

SID

Session ID of the lock list owner

The values indented under individual lock lists are lock block values:

Id

Internal Lock block identifier

Rsb

Internal Resource block identifier

Gr

Granted lock mode

Req

Requested lock mode

State

Current state of lock (GR = granted, WT = waiting)

KEY

Information used to identify the resource being locked.

- When checking contention on data pages, the key will contain PAGE, the database ID, the table reltid and reltidx, and the page number.
- ROW is a special type of lock used to reserve space for deleted rows in four core catalogs only: iirelation, iirel_idx, iiattribute and iidevices.

Lockstat Command Output – Locks by Resource

The "Locks by resource" portion of lockstat groups the individual locks by resource block and shows any contention that can lead to query performance problems.

Here is sample output from this section:

```
--------------------------Locks by resource-----------------------
Id: 00000020 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000002>
     KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=3)
  Id: 0000000A Llb: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000021 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
     KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=30)
  Id: 0000000B Llb: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000022 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
     KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=2)
  Id: 0000000C Llb: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000023 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
     KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=23)
  Id: 0000000D Llb: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000024 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
     KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=1)
  Id: 0000000E Llb: 0000000B Gr: IS Req: IS State: GR PHYS(1)
Id: 00000025 Gr: IS Conv: IS Cbacks 0 Value: <0000000000000000>
     KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=33)
  Id: 0000000F Llb: 0000000B Gr: IS Req: IS State: GR PHYS(1)
```

Fields are as follows:

Id

Internal Resource block identifier

Gr

Granted mode of the resource

Cbacks

Number of resource locks that contain DMCM callback information. This field is relevant only when the installation has been configured to run with the Distributed Multi-Cache Management (DMCM) protocol.

Conv

Conversion mode requested on the resource

Value

Lock value associated with the resource

KEY

Byte string identifying the resource

The indented portions of the resource blocks show the individual lock blocks that are contending for the resource. These lock blocks are described as follows:

Id

Internal Lock block identifier

Llb

Lock list identifier on which this lock resides

Gr

Granted mode of the lock

Req

Requested lock mode

State

Current state of the lock: GR is granted; WT is waiting

Lockstat Command Output – DLM Locks

The DLM Locks portion of the lockstat utility displays the cluster DLM (Distributed Lock Manager) view of the Ingres installation locks (grouped by resource).

Here is sample output from this section:

```
--------------- DLM Locks taken by Ingres instance CL ---------------

Locks by Resources having blocked lock(s)
  RqstNode           PID      Lock ID    Q GR RQ (MastNode:MastID)
LK_CSP,NODE=1,PID=0
  NODEA            2464D19B   47000C37 - G EX EX (NODEA:47000C37)
  NODEA            2464D19B   5B002C55 - W NL PR (NODEA:5B002C55)
  NODEB            24C34D01   4900C1FD - W NL PR (NODEA:61004665)

LK_CSP,NODE=1,PID=2464a9bb
  NODEA            2464A9BB   43006E07 - G EX EX (NODEA:43006E07)
  NODEA            2464D19B   7400381B - W NL PR (NODEA:7400381B)
  NODEB            24C34D01   0A006AED - W NL PR (NODEA:240057A6)

LK_CSP,NODE=2,PID=0
  NODEB            24C34D01   5900143D - G EX EX (NODEB:5900143D)
  NODEB            24C34D01   1900B776 - W NL PR (NODEB:1900B776)
  NODEA            2464D19B   2C000B37 - W NL PR (NODEB:46005BA6)

LK_CSP,NODE=2,PID=24c34d03
  NODEB            24C34D03   640055EB - G EX EX (NODEB:640055EB)
  NODEB            24C34D01   110038B8 - W NL PR (NODEB:110038B8)
  NODEA            2464D19B   22003414 - W NL PR (NODEB:4700B1B5)

LK_CSP,NODE=2,PID=24c34d04
  NODEB            24C34D04   6700CA0C - G EX EX (NODEB:6700CA0C)
  NODEB            24C34D01   7A00C422 - W NL PR (NODEB:7A00C422)
  NODEA            2464D19B   21005449 - W NL PR (NODEB:5700C08D)

Cluster Nodes seen
  Node Name     Node Number     Lock Count
  Local          00000000              0
  NODEA          00010023             13
  NODEB          00010026             17
```

Fields are as follows:

RqstNode

Node from which the lock was requested

PID

Process ID

Lock ID

Lock identifier

Q

Status of the lock:

G–granted

W–waiting

C–convert

GR

VMS lock mode granted:

NL–NULL

CR–Concurrent Read

CW–Concurrent Write

PR–Protected Read

PW–Protected Write

EX–Exclusive

RQ

VMS lock mode request (See GR above for lock mode explanations.)

MastNode:MastID

Master node and master lock ID

logstat Command—Display Logging Status

Permission required: Installation owner.

The logstat utility displays logging status. Logstat output has the following sections:

- Logging System Summary
- Current log file header
- List of active processes
- List of active databases
- List of active transactions

Note: Logstat functions are included in the forms-based Interactive Performance Monitor (ipm) utility. Also, you can use Visual DBA to monitor log information. See Visual DBA online help.

For more information on the information displayed by the logstat command, see the *Interactive Performance Monitor User Guide*.

The logstat command has the following format:

```
logstat [-buffers] [-databases] [-header] [-statistics]
[-processes] [-transactions] [-user_transactions]
[-special_transactions] [-all_transactions] [-verbose] [-help]
dynamic
```

-buffers

Displays list of log buffers and their usage

-databases

Displays information on databases connected to the logging system

-header

Displays current log file header

-statistics

Displays logging system summary

-processes

Displays a list of processes connected to the logging system

-transactions

Displays information on all transactions (same as -all_transactions)

-user_transactions

Displays information on user transactions

-special_transactions

Displays information on system transactions

-all_transactions

Displays information on all transactions (same as -transactions)

-verbose

Displays all information

-help

Displays command options online

dynamic

Displays a line of the current log information to a command window every few seconds

Logstat Command Output – Logging System Summary

The Logging System Summary on the logstat output provides an overall view of the logging system, indicating how well the logging system is tuned.

The summary contains the following informational fields (described in the order in which they appear):

Database add

Indicates the number of times a database has been added to the logging system. This number is incremented whenever a session is the first session to access the database.

Database removes

Indicates the number of times a database has been removed from the logging system. This number is decremented when the last session accessing the database exits.

Transaction begins

Indicates the number of times a transaction has been started in the logging system. This number indicates the total number of transactions started in the logging system.

Transaction ends

Indicates the number of times a transaction has been completed in the system during normal processing. This number corresponds to all the transactions that were committed or rolled back without incident and properly terminated. This value does *not* include FORCE-ABORT, LOGFULL, or any other rcp actions that terminated a transaction abnormally.

Log read i/o's

Indicates the number of times a read was performed on the log file. This is a physical operation. This number corresponds to the number of times the log buffer was read from the transaction log file to perform an abort, archive or purge operation.

Log write i/o's

Indicates the number of times a write was performed on the log file. This is a physical operation. This number corresponds to the number of times a log buffer was written to the transaction log file.

Log writes

Indicates the number of writes from the database buffer into the log buffers. These are memory-to-memory writes.

Log forces

Indicates the number of requests made to the logging system to force the current log buffers to the log file. This is most frequently done to commit a transaction or to guarantee the consistency of the log file before writing an update to the database.

Log optimized writes

Indicates the number of times the logging system was able to combine multiple log pages in a single physical write. Optimized writes are enabled by the CBF parameter "ii.*.rcp.log.optimize_writes: ON".

Log optimized pages

Indicates the number of pages written by log optimized writes.

Log waits

Indicates the number of times any event wait condition requires a log buffer write to stall. These are events such as LOGFULL, CP writing, RECOVERY, ARCHIVING required, FREE WAIT for log buffers, OPENDB wait, log buffer SPLIT wait, wait for completion of log i/o (that is, from the log buffer to the log file). "Log waits" is the sum of "log waits by type."

Log splits

Indicates the number of log records that spanned two or more buffers. Log splits in and of themselves are not to be interpreted as bad events. What is potentially harmful is the inability of the logging system to proceed with the log record split. This situation can be remedied by adding additional log buffers to the system or by increasing the size of the current buffers to minimize the need for splits. Any modification should be examined in conjunction with the effect that it has on the other wait states.

Log group commit

Indicates the number of times that multiple transactions participate in a log buffer flush to the log file.

Log group count

Indicates the number of transactions that are participating in the flush to the log file. This value is the wait count associated with the Log group commit count above. This value is incremented based on the number of waiters at write completion time.

The ratio of Log group count to Log group commit gives an indication of how effectively the group commit mechanism is working in the current configuration.

Check commit timer

Indicates the number of times the timer associated with the group commits completes. This does *not* necessarily mean that a write to the log file has to occur, because the log buffer that initiated the timer may have already been written due to being full.

Timer write

Indicates the number of times a log file write actually occurs because of the timer expiration or inactivity in the log buffer. As explained above, this will occur only if the buffer has not completely filled before this timer expires.

Timer write, time

Indicates the number of times those writes were due to the expiration of the group commit timer.

Timer write, idle

Indicates the number of times those writes were due to inactivity in the buffer.

Inconsistent db

Indicates the number of times the logging system has had to mark a database inconsistent due to an inability to recover some portion of work that currently exists in the logging system.

Kbytes written

Indicates the number of bytes written to the log file

ii_log_file read

Indicates the number of physical reads of the primary log file

ii_dual_log read

Indicates the number of physical reads of the dual log file

Write complete

Indicates the number of times a write of the primary log file completes successfully

Dual write complete

Indicates the number of times a file write of the dual log file completes successfully

All logwriters busy

Indicates the number of times that a log buffer needed to be written but no idle logwriter in any DBMS server could be found to write it. This causes a log wait until a logwriter thread is available.

The statistics All logwriters busy, Max write queue len, and Max write queue cnt represent activity of the log writers.

The "All logwriters busy" count should be a small fraction of overall log waits (10% or less). A high count indicates a lack of logwriter threads relative to the number of log buffers. If increasing the number of logwriter threads does not help, look for a disk or controller bottleneck writing to the transaction log.

Max write queue len

Indicates the maximum number of log buffers queued for a physical log write.

The "Max write queue len" value is usually equal to the number of log buffers, unless the system is configured with more log buffers than necessary.

Max write queue cnt

Indicates the number of times the log write queue length reached "Max write queue len."

The "Max write queue cnt" value indicates how busy the logging system is. A high value in conjunction with significant numbers of log split waits or log free waits may indicate a lack of log buffers, or a physical disk I/O problem with the transaction log.

Log Waits By Type

Shows log wait totals categorized by the type of event that caused the wait:

Force

Indicates the number of times a thread was stalled waiting for a log force operation to complete.

Free Buffer

Indicates the number of times all the log buffers are either in force mode or unavailable for writing. One log buffer is written to at a time. If free buffer waits are frequent, then increasing the number of log buffers may be the solution. Remember that an increase in the number of buffers requires (number_of_log_buffers * log_buffer_size) requires more memory on the host system.

Split Buffer

Indicates the number of times a thread was stalled waiting for a log split operation to complete.

Log Header I/O

Indicates the number of times a thread was stalled waiting for a log header I/O operation to complete.

Ckpdb Stall

Indicates the number of times a thread was stalled waiting for a ckpdb (checkpoint a database) operation to complete.

Opendb

Indicates the number of times a thread was stalled waiting for an open database operation to complete.

BCP Stall

Indicates the number of times a thread was stalled waiting for Begin Consistency Point information to be written to disk. This is a very brief stall performed at the start of a consistency point.

Logfull Stall

Indicates the number of times a thread was stalled waiting for a LOGFULL condition to clear. All logging system writes are stalled for users until the condition is cleared. This is most often seen from the user's viewpoint as a "hung" system. Always check logstat for the status in the header block. If this value is LOGFULL then this is a stall condition.

Lastbuf

Indicates the number of times a thread was stalled waiting for its COMMIT record to be written.

Forced I/O

Indicates the number of times a thread was stalled waiting for a forced I/O operation to complete.

Event

Indicates the number of waits for internal non-I/O logging operations.

Mini transaction

Indicates the number of waits for internal transactions to complete.

Logfull Commit

Indicates the number of times a thread was stalled in conjunction with SET SESSION WITH ON_LOGFULL = COMMIT protocols.

Logstat Command Output – Current Log File Header

The Current Log File Header gives quantitative information on the logging system, such as the size of the log file, log buffers, and CP interval.

Consistency Points (CP)—In the Current log file header section is a group of numbers preceded by the label CP. These numbers, like the numbers following Begin and End, are in three groups. The middle group refers to the block marking the last consistency point. This consistency point contains a list of all open transactions and open databases at that time.

In the sample output shown here, the block marking the consistency point is 13909. CPs shorten the recovery window after a system goes down. Instead of reading from BOF to EOF, the last CP is read and recovery begins from there.

```
----Current log file header-----------------------------------------------------
Block size: 4096  Block count: 2048  Partitions: 1  Buffer count: 4
CP interval: 102  Logfull interval: 1945  Abort interval: 1536
Last Transaction Id:  00002D5B2D5BFA03   Last LSN: <1187966476, 1054245>
Begin: <1187966477:13909:2968> CP: <1187966477:13909:2968>
End: <1187966477,13909,1012>
Forced LGA,LSN: <1187966477,13909,1012>,<1187966476,1054245>
Percentage of log file in use or reserved: 6
Log file blocks reserved by recovery system: 180
Archive Window: <0,0,0>..<0,0,0>
Previous CP:  <0,0,0>
Status:     ONLINE,ARCHIVE,CPFLUSH
Active Log(s):     II_LOG_FILE
```

The Current Log File Header section has the following fields:

Block size

Indicates the size of the log buffer and log file blocks in bytes. The log file is organized as a series of blocks that are laid down in a circular fashion and used for on line backup.

Block count

Indicates the size of the log file in blocks

Partitions

Indicates the number of log partitions in the log file

Buffer count

Indicates the number of log file buffers. All processes connected to the logging system share the buffers.

CP interval

Indicates the number of blocks between consistency points. CPs may also be caused by other events, such as archiver PURGEs and online checkpoints.

Logfull interval

Indicates the number of log file blocks used before LOG_FULL is signaled

Abort interval

Indicates the number of log file blocks that must be used before a FORCE_ABORT is signaled

Last Transaction Id

Indicates the ID of the last transaction to write a log record

Last LSN

Indicates the log sequence number associated with the last written log record

Begin, CP, End

Indicates the log addresses of the beginning of the log file, the last consistency point, and the end of the log file

Forced LGA,LSN

Indicates the LGA (physical log file address) and LSN of the last log record written to the log file

Percentage of log file in use or reserved

Indicates the percentage of the log file that has either been used or is reserved for use by the recovery system

Log file blocks reserved by recovery system

Indicates the number of log file blocks reserved for transaction recovery operations. Space reserved by a transaction is freed when the transaction commits normally, or it is used to write compensation log records during transaction abort processing.

Archive Window

Indicates the segment of the log file that can be examined by the archiver for journal or dump processing

Previous CP

Indicates the log file address of the last consistency point. This is the position in the log file where the last consistency point was taken.

Status

Indicates the current logging system status. Status can be one or more of the following values:

ACP_SHUTDOWN—the archiver is preparing to shut down. (This indicates that an rcpconfig command with the shutdown option has been issued.)

ARCHIVE—the archiver process is archiving journaled transactions to the journal files.

BCPSTALL—the logging system is requesting the recovery process to start writing a begin consistency point.

CKP_SBACKUP—the logging system marks the start of online backup. It marks this block as the online backup start block (SB). Ckpdb starts backing up the database.

CLOSEDB—the logging system is in the process of closing a database.

CPFLUSH—DBMS Servers are flushing their modified pages to disk, that is, a consistency point is being taken.

CPNEEDED—the logging system is about to take a consistency point.

CPWAKEUP—the logging system is synchronizing the fast-commit threads.

DISABLE_DUAL_LOGGING—the logging system is in the process of disabling dual logging.

DUAL_LOGGING—dual logging is enabled. (Note that this does not mean that both primary and dual logs are active. For active logs look at the Active Log(s) field.)

ECP—the logging system is requesting that the recovery process start writing an end consistency point.

ECPDONE—the logging system has taken an end consistency point. This status flag is present most of the time while the logging system is functioning normally.

FORCE_ABORT—the force-abort-limit has been reached; the oldest open transaction will be aborted.

IMM_SHUTDOWN—the logging system has been told to shut down immediately. (This is displayed when the user invokes rcpconfig with the imm_shutdown option.) Note that the logging system does not perform any housekeeping as part of the shutdown process. The recovery process then becomes responsible for backing out any uncommitted transactions left in the log file once the logging system has been restarted.

JSWITCHDONE—the logging system has completed a switch of the journal file. This status flag is present most of the time while the logging system is functioning normally.

LOGFULL—the log file is full. The system administrator should determine the cause of this and increase the log file size. A warning indicator is also displayed.

MAN_ABORT—the logging system has been requested to manually abort a distributed transaction.

MAN_COMMIT—the logging system has been requested to manually commit a distributed transaction.

ONLINE—the logging system is on line. The logging and recovery systems are operating OK.

OPENDB—the logging system is in the process of opening a database.

PURGEDB—a database has been closed by the last user who had it open; the archiver is archiving transactions that belong to this database.

RCP_RECOVER—the recovery process is recovering transactions from a runaway DBMS.

RECOVER—the logging system has requested the recovery process to perform recovery.

START_ARCHIVER—this important status indicates that the archiver has stopped and must be restarted by the DBA. *This is not done automatically.* If the DBA does *not* do it, the log file will eventually fill up, reaching the LOG_FILE_FULL limit, and cause the system to stall.

START_SHUTDOWN—the logging system is shutting down. As part of the shutdown process, the logging system commits to disk all the committed transactions and backs out any uncommitted ones. The archiver also journals all the committed transactions for tables with journaling enabled.

Active Log(s)

Displays log files that are active.

Logstat Command Output – List of Active Processes

The List of Active Processes provides information on processes currently active in the logging system. This section has the following fields:

ID

Indicates the internal logging system ID for a process

PID

Indicates the process ID

TYPE

Indicates the type of the active process. The type field can be:

FCT—a DBMS Server running Fast Commit

SLAVE—a DBMS Server not running Fast Commit

ARCHIV—the archiver process

MASTER—the recovery process

OPEN_DB

Indicates the number of times the server opened a database. The recovery process and archiver each have their own database opened at all times, while the server is the process that opens the databases.

WRITE

Indicates the number of writes this process has performed in the logging system

FORCE

Indicates the number of times this process requested that a log buffer be forced to disk

WAIT

Indicates the number of times any transaction in this process needed to wait for a logging system-related reason

BEGIN

Indicates the number of transactions started by this process

END

Indicates the number of transactions ended by this process

Logstat Command Output – List of Active Databases

The List of Active Databases provides statistical information on all active databases in the logging system. You can use the information in this section and the List of Active Transactions section to determine which databases are open and active. Before shutting down an installation, you should ensure that all databases are closed. Knowing which databases are open and active allows you to determine whom to notify of the impending shutdown.

This section has the following fields:

Id

Identifies the logging system ID number of an active database

Database

Identifies the name of the database and of the DBA.

Status

Indicates the current state of the database.

Tx_cnt

Indicates the number of transactions currently active in the database

Begin

Indicates the number of transactions started in this database

End

Indicates the number of transactions ended in this database

Read

Indicates the number of reads that the logging system performed on behalf of this database

Write

Indicates the number of writes that the logging system performed on behalf of this database

Force

Indicates the number of times that the logging system had to force out the log buffer on behalf of this database

Wait

Indicates the number of times that the logging system had to wait for a log buffer on behalf of this database

Location

Indicates the physical location of this database in the file system

Journal Window

Indicates the active journal window on this database. If no journaling is active on the database, the window has boundaries <0,0,0>. .<0,0,0>.

Start Backup Location

Indicates the log file end-of-file (EOF) address when a database backup is started. This address is used during online backup processing.

Logstat Command Output – List of Transactions

The List of Transactions provides statistical information on each active transaction. This section has the following fields:

Tx_id

Identifies the transaction ID used by the logging system

Tran_id

Identifies the transaction. The transaction ID is used by both the logging and locking systems. The ID is useful when you want to follow a transaction from lockstat to logstat output.

Database

Identifies the ID of the database. This ID is the same as the ID in the List of active databases section of the logstat output.

Process

Indicates the ID of the process currently working on this transaction. This field corresponds to the internal logging system ID in the List of active processes section of logstat.

Dis_tran_id

Currently not used

Session

Indicates the user session ID that owns this transaction. This is the same ID used in iimonitor output. Use this ID to locate the user and the terminal that initiated the transaction.

First

Indicates the log file address of the first record associated with this transaction

Last

Indicates the log file address of the last record associated with this transaction

Cp

Indicates the first consistency point address taken that concerns this transaction

FirstLSN

Indicates the log sequence number associated with the first record written to the log file by this transaction

LastLSN

Indicates the log sequence number associated with the last record written to the log file by this transaction

WaitLSN

Indicates the log sequence number that must be written to the log before the transaction can proceed

Write

Indicates the number of log buffer writes because of this transaction

Split

Indicates the number of times this transaction had to wait for a log buffer in order to write a log record that spanned multiple buffers

Force

Indicates the number of times the log buffer was flushed. The force conditions are commented in more detail under the Log forces field in the Summary section above.

Wait

Indicates the number of times this transaction had to wait for a logging system-related event

Reserved

Indicates the number of log blocks reserved by this transaction for recovery operations

WaitBuf

Indicates the log buffer on which the transaction is waiting to be written to the log

Status

Indicates the status of this transaction. This field can take the following values:

ACTIVE—this transaction has written a number of records to the log file.

INACTIVE—this transaction is in the retrieve mode and has not written any records to the log file.

PROTECT—this transaction is a user transaction (as opposed to an internal system transaction) and will be recovered in the event of a server or system failure.

JOURNAL—this transaction must be journaled. This flag indicates that the transaction should be archived.

Wait Reason

Indicates why the transaction is waiting. Wait Reason can have the following values:

(not waiting): The transaction is not waiting.

FORCE—waiting for a log force

FREE—waiting for a free log buffer

SPLIT—waiting for a log split completion

HDRIO—waiting for log header I/O completion

CKPDB—waiting for a ckpdb completion

OPENDB—waiting for an open database completion

BCPSTALL—waiting for BCP log write to complete

LOGFULL—waiting because of LOGFULL condition

FREEBUF—waiting for a free buffer

LASTBUF—waiting for the last buffer in the transaction to be written

BUFIO—waiting for a log buffer to be freed

EVENT—waiting for a log event

ABSOLUTE_LOGFULL—waiting at the absolute end of a LOGFULL condition

User

Indicates the owner of this transaction. User can have the following values:

logfile_I/O_thread—log file read/write thread

group_commit_thread—group commit thread

buffer_manager—the buffer manager

log_reader_transaction—log file read/write thread

recovery_thread—the DMFRCP recovery thread

consistency_pt_thread—the consistency point thread

consistency_point_timer—the consistency point timer thread

write_behind—the write behind thread

security audit thread—in C2 enabled systems only, the security audit thread

username—user session

logstat Example: Determine Databases that Are Active

To determine which databases are currently active, follow these steps:

1. Look at the List of active databases in the logstat output. For each database listed, the database's ID number, name, owner, and status appear on one line.

   Note: The first entry listed is always owned by $ingres.

2. Note the ID number of each database. These numbers are used in the List of active transactions in the logstat output to identify the database associated with each transaction.

3. Compare the database ID numbers to the entries following the heading Database in the listings of active transactions. If you find a match, the database associated with that ID is currently active.

For example, in the sample output shown here, the second database shown is testdb owned by test. The ID for this database is 00280005. In the List of active transactions, the transaction listed belongs to Database: 00280005. The status of this database (testdb) is ACTIVE,PROTECT,JOURNAL. If an installation shutdown was pending, you can inform the testdb's owner, test, of the impending shutdown.

```
----List of active databases-----------------------------------------
Id: FFFF0001  Database: ($recovery,$ingres)  Status: NOTDB,ACTIVE
  Tx_cnt: 13 Begin: 16 End: 1 Read: 0 Write: 217 Force: 583 Wait: 1707
  Location:   None
  Journal Window: <0,0,0>..<0,0,0>
  Start Backup Location: <0,0,0> (0,0)
Id: 00280005  Database: (testdb,test)  Status: JOURNAL,FAST_COMMIT,ACTIVE
  Tx_cnt: 1 Begin: 17 End: 16 Read: 0 Write: 4690 Force: 6 Wait: 681
  Location:   /devsrc/65sun4/install/test/ingres/data/default/testdb
  Journal Window: <760996814,981,2800>..<760996814,1300,3184>
  Start Backup Location: <0,0,0> (0,0)

----List of active transactions-------------------------------------
  [... transaction information deleted ...]

Tx_id: 295D001D  Tran_id: 00002D5B2D5BF9F7  Database: 00280005
  Process: 00010012  Dis_tran_id: <0,0> Session: 0093C000
  First: <760996814,1160,3100>  Last: <760996814,1300,3184>
    Cp: <760996814,881,948>
    FirstLSN: <?,?>  LastLSN: <?,?>  WaitLSN: <?,?>
  Write: 463 Split: 107 Force: 0 Wait: 109 Reserved: 140
  Status: ACTIVE,PROTECT,JOURNAL
  Wait Reason: (not waiting)
  User: <test>
```

logstat Example: Determine Proximity to FORCE-ABORT-LIMIT

To determine how close your installation is to the FORCE-ABORT-LIMIT, use the information from the Current log file header section in the logstat output. Four statistics are relevant: the Abort interval, the Block count, the Begin, and the End.

The number appearing after Abort interval is the number of blocks in the log file that must be filled before the FORCE-ABORT-LIMIT is reached. The Block count refers to the total number of blocks in the log file. Begin refers to the block marking the log file's Beginning of File (BOF), and End refers to the block marking the log file's End of File (EOF). The numbers following Begin and End are divided into three groups separated by colons. The middle group is the most relevant. For example, in the sample output, block 778 marks the beginning of the log file.

To calculate how close the installation is to the FORCE-ABORT-LIMIT, follow these steps:

1. Calculate the number of blocks in use.

   Because the log file is a circular file, the block marking the file's beginning can have a higher number than the block marking the file's end (see the Second Log File Header).

   Consequently, there are two ways to determine the number of blocks in use:

   a. If the End of File is larger than the Beginning of File, subtract the BOF from the EOF to obtain the number of blocks in use. For example, in the sample output, the BOF is 778 and the EOF is 1299. The number of blocks in use in this example is:

      1299 - 778 = 521

   b. If the End of File is smaller than the Beginning of File, subtract the BOF from the block count figure and add the result to the EOF to obtain the number of blocks in use. For example, in the second Log File Header, the Beginning of File is 1702, the Block Count is 2048, and the End of File is 107. The number of blocks in use is:

      (2048 - 1702) + 107 = 453

2. Subtract the number of blocks in use from the Abort interval figure to determine how many blocks are available before the FORCE-ABORT-LIMIT is reached.

For example, in the sample output shown here, 521 blocks are in use and the Abort interval is 1536, so the number of blocks still available is:

1536 - 521 = 1015

```
----Current log file header---------------------------------

Block size: 4096  Block count: 2048  Partitions: 1  Buffer count: 4
CP interval: 102  Logfull interval: 1945  Abort interval: 1536
Last Transaction Id:  00002D5B2D5BFA03  Last LSN: <760996813,1054245>
Begin: <760996814:1702:2304> CP: <760996814:1873:3592>
End: <760996815:107:20>
Forced LGA,LSN: <760996814,107,20>,<760996813,1054245>
Percentage of log file in use or reserved: 30
Log file blocks reserved by recovery system: 180
Archive Window: <760996814,1991,3508>..<760996815,107,220>
Previous CP:  <760996814,1702,2304>
Status:     ONLINE,ARCHIVE,CPFLUSH
Active Log(s):     LOG_FILE
```

mkrawarea Command—Make a Raw Area File

Valid on UNIX.

Permission required: You must run this utility as root.

The mkrawarea command is run at installation time from the root login to create a raw database area and link it to a character special file.

The mkrawarea command has the following format:

```
mkrawarea
```

mkrawlog Command—Make a Raw Log File

Valid on UNIX.

Permission required: You must run this utility as root.

The mkrawlog command is run at installation time from the root login to create a raw log file for the transaction log. You can optionally run this command after installation to set up a raw log file for the dual log.

The mkrawlog command has the following format:

```
mkrawlog [-dual]
```

mkrc Command—Have Ingres Start with Operating System

Valid on Linux.

The mkrc utility generates and installs an RC script to have Ingres start up and shut down with the operating system (Linux only). The generated script is called ingresXX, where XX is the installation ID, and is placed in $II_SYSTEM/ingres/files/rcfiles.

The mkrc command has the following format:

```
mkrc [-i [123456]]|[-r]
```

-i

Installs the RC script under /etc/init.d. If a run level of 1 or more is specified, links to the corresponding RC directories are created. If no run level is specified, links are created under the levels defined by 235.

-r

Removes script from /etc/init.d.

modifyfe Command—Modify Storage Structure of Catalog

The modifyfe command modifies the storage structure of catalogs for Ingres querying and reporting tools such as Vision, Applications-By-Forms, or OpenROAD. For more information on these user interface catalogs, see the *Database Administrator Guide.*

The modifyfe command (like upgradedb) takes an exclusive lock on the database.

Typically, you do not issue the modifyfe command directly; it is called by the upgradedb utility to perform the required catalog updates. For information on using upgradedb and modifyfe to upgrade the installation, see the *Migration Guide*.

The modifyfe command has the following format:

```
modifyfe dbname |vnode::dbname[/server_class] [-uusername] [+w|-w] {product}
```

dbname

Specifies the name of the database containing the catalogs to be modified, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-u*username*

Specifies the effective user, as described in Standard Flags and Parameters (see page 13). If you want to modify a database you do not own, you must use this flag to specify the user name of the DBA.

+w|-w

Waits (+w) or does not wait (-w) for the database to be free (not in use). The default is -w.

If you specify -w or if you do not specify this flag, then modifyfe aborts immediately if the database is not free (that is, another session has an exclusive connection).

If you specify +w then modifyfe waits for anyone with an exclusive connection to disconnect, then proceeds.

product

Specifies the products for which you want to modify catalogs. Valid product names are ingres, ingres/dbd, vision, and windows_4gl, as described in Standard Flags and Parameters (see page 13).

If you omit this parameter, all user interface catalogs are modified.

netutil Command—Start Network Management Utility

The netutil command invokes the Net Management Utility, a forms-based program for configuring Ingres Net. This command allows any number of Ingres sites to be connected together in a single network.

The Net Management Utility allows you to store and manage the information—connection data and remote user authorizations—needed by the Communications Server and Bridge Server to connect to remote installations.

The netutil command has the following format:

```
netutil [-u user] [-vnode vnode] [-file filename{,filename}]
```

-u *user*

Specifies the effective user name for the session. When creating private connection information, the information will be stored for the specified user.

-vnode *vnode*

Identifies the name of the remote node on which the connection information is to be stored. This vnode name must have been configured previously through either the netutil or ingnet utility.

-file *filename*{,*filename*}

Operates netutil non-interactively. All statements in the specified control file are executed.

When the input file name is specified as - (a single dash character), input is taken from the standard input channel. This allows the user to enter commands directly from the keyboard.

netutil Examples

This command edits private connection information for the user emma:

```
netutil -uemma
```

This command edits connection information for the previously defined node new_york:

```
netutil -vnode new_york
```

This command runs netutil in interactive mode, taking input from the keyboard, for the user emma on the remote node new_york:

```
netutil -uemma -vnode new_york -file-
```

optimizedb Command—Generate Statistics for the Query Optimizer

The optimizedb command generates statistics that are used by the Ingres Query Optimizer to select an efficient query processing strategy.

Statistics are generated on the specified columns, and stored in system catalogs (iistats and iihistograms).

Complete and accurate statistics in the system catalogs result in more efficient query execution strategies and faster system performance. The process of generating complete and accurate statistics requires time, but a balance between accurate statistics and the time to generate them can be achieved by specifying the -zx or -zs flag. Statistics need to be refreshed only when a significant change in the distribution of a column's values has occurred.

The statistics generated by the optimizedb command for any column consist of two elements:

1. The number of unique values in a column
2. A histogram with a variable number of variable-width cells

The accuracy of the histograms can be controlled by the -zu# and -zr# flags described below. Increasing the number of cells in the histograms increases the amount of space required for the iihistograms table and thus increases somewhat the amount of space and time used by the optimizer. However, the increased accuracy of the statistics generally results in more efficient query execution strategies.

Note: By default, optimizedb uses sampled statistics for tables that have more than 500,000 rows.

We recommend that you generate the statistics for all columns that appear in the qualification (where clause) of a query statement. If statistics are missing or incorrect, the query will still execute, but the speed of query processing can be affected.

After running optimizedb, it is prudent to run sysmod. This is especially true the first time optimizedb is run on a database.

Note: Although optimizedb does not lock the database or individual tables while it is retrieving values and generating statistics, after the statistics have been collected and stored in the appropriate catalogs, optimizedb takes an exclusive lock on the database or individual tables to complete its task.

For additional information on the Ingres Query Optimizer and the use of the optimizedb command, see the *Database Administrator Guide.*

The optimizedb command has the following format:

```
optimizedb [SQL option flags][-i filename] [-o filename] [-z flags]
dbname[/server_class] {-r tablename {-a columnname}} | {-xr tablename} [-help]
```

SQL option flags

Indicate SQL option flags that are automatically passed. The optimizedb command accepts the following SQL option flags. For a complete description of these flags, see the sql Command (see page 245).

+U | -U

-u

-c*N*

-t*N*

-i*kN*

-f*kxM.N*

+w | -w

-x*k*

-i *filename*

Reads statistics from *filename* instead of operating directly on the database.

The *filename* must be a file in ASCII format that was generated by the statdump command using the -o flag. While you can edit this file, only two types of changes are acceptable: modifying values and adding rows that describe cells. Do not change the format of the file, that is, do not change the order in which data appears or add an incomplete new row.

The -r and -a flags, when used with this flag, act as filters. Optimizedb reads in from the file only those statistics that belong to the specified table or column.

Optimizedb does not use the row and page count values in the file unless the -zp flag is also specified.

Note: These values are vital for correct operation of the DBMS. If you use the -zp flag, be sure to put new values for row and page counts in iitables.

Warning! A file with histogram data represented in hex format (generated by the -zhex flag) cannot be used as input to the optimizedb -i command. Doing so will result in incorrect histogram data, which will affect the performance of optimization algorithms.

-o *filename*

Writes the output to the specified file instead of to the system catalogs.

-z flags

Specify options to optimizedb. For details, see Optimizedb -z Flags (see page 195).

dbname

Indicates the name of the database, and if required, the *server_class*, as described in Standard Flags and Parameters (see page 13).

-r*tablename*

Specifies *tablenames* to be processed. If no table name is specified, then all columns for all tables in the database are processed.

The table name can be qualified with a valid schema name in the format *schema*.*tablename*, as described in Schema Qualifier (see page 18).

If *tablename* specifies a secondary index name, optimizedb creates a composite histogram on the key columns comprising the index.

-xr*tablename*

Specifies one or more *tablenames* to be excluded from processing. Except for these tables, all columns in all tables in the database are processed.

Note: Using both the -r*tablename* and -xr*tablename* parameters is not permitted in a single optimizedb request; nor is using both the -xr*tablename* and -a*columnname* parameters.

-a*columnname*

Limits processing to the specified columns plus any columns included through the -zk flag. You can use the -a*columname* flag **only** if the -r*tablename* parameter is specified.

-help

Displays command syntax online.

Optimizedb -z Flags

The –z flags on the optimizedb command are as follows:

-zc

Directs optimizedb to optimize the system catalogs in addition to the base tables. If you want to optimize selected system catalogs only, use this flag and specify the individual tables with the -r flag. This flag is valid only if the user issuing the command is the DBA for the specified database.

-zcpk

Requests a composite histogram on primary key structure.

-zdn

Directs optimizedb to use its algorithm to estimate the number of distinct values and repetition factor for a column whose histogram is built with sampling (see the -zs# option).

-zf*filename*

Directs optimizedb to read *filename* for all other command line flags, database names, and any other command line arguments. This file must contain only one flag per line (see the examples below). If this flag is specified, no other flags or arguments can appear on the command line; they must, instead, appear in the specified file.

-zfq

Directs optimizedb to use the "fast query" option, which significantly reduces the time to build a histogram. This option improves performance only when the repetition factor of the column is 20 or higher.

The -zfq flag can also cause optimizedb to generate a global temporary table from the values of the histogrammed columns when more than one column is identified in the optimizedb command. The histograms are then built by reading from the faster temporary table, rather than from the base table. The smaller and faster temporary table offers additional performance benefits for the fast query option.

Optimizedb builds the global temporary table when -zfq is specified, and when the number of histogrammed columns and the size of the temporary table row (relative to the size of a base table row) meet certain criteria. See the description of the –znt flag, which can be used with the –zfq flag.

Because there is no performance benefit in building more than one histogram on a table with a single execution of optimizedb, it is recommended that repetitious columns be specified in one execution of optimizedb (with the –zfq flag) and that the others be specified in a separate execution.

-zh

Prints the histogram that was generated for each column. This flag also implies the -zv flag.

-zhex

Generates histogram cell values in hex format, which is useful for seeing how Unicode data is stored. This flag is only effective when used with the -zh and the -o flags.

-zk

Generates statistics for columns that are keys on the table or are indexed, in addition to columns specified on the command line.

-zlr

Reuses existing repetition factor if there is one.

-zns

Disables the default behavior of creating histograms from a maximum 500,000 row sample. Using this parameter assures that all rows are read from a table during the histogram building process.

-znt

Disables the use of global temporary tables when using the "fast query" option (-zfq) if disk space is not sufficient.

This flag is used only with the -zfq flag.

-zn#

Directs optimizedb to read floating-point numbers using the precision level specified by #. Use this flag in conjunction with the -i *filename* flag.

-zp

Directs optimizedb to read the row and page count values in the file specified with the -i flag and to store those values in the appropriate system catalog (they can be viewed in iitables).

-zr#

Specifies the maximum number of cells that the histogram can contain if optimizedb creates an inexact histogram. In an inexact histogram, each cell represents a range of values.

The allowable range is 1<#<15000 (that is, the minimum is 2 and the maximum is 14999).

The default number of cells is 100.

-zs[s]#

Creates statistics based on sample data. The percentage of table rows sampled is determined by the value of #. This number must be a floating-point number in the range of 0 to 100. Specifying the optional s (-zss) will cause the tuple identifiers (TIDs), which are used to retrieve the sample rows, to be sorted before the rows are retrieved. This decreases retrieval time but increases the amount of memory used by optimizedb.

-zu#

Specifies the maximum number of cells an exact histogram can contain. In an exact histogram, each cell represents a single, unique value.

The allowable range is 1 to 15000.

The default number of cells is 100.

-zv

Prints information about each column as it is being processed.

-zw

Sets the complete flag, which indicates whether a column contains all possible values. The range of values in a column affects query optimization. By default, columns are assumed to be not complete.

-zx

Directs optimizedb to determine only the minimum and maximum values for each column rather than full statistics. Because minimum and maximum values for columns from the same table can be determined by a single scan through the table, this flag provides a quick way to generate a minimal set of statistics. Minimal statistics cannot be created on columns holding only null values.

optimizedb Example: Generate Full Statistics for a Database

This command generates full statistics for all columns in all tables in the empdata database:

```
optimizedb empdata
```

optimizedb Example: Generate Statistics for Certain Columns

This command generates statistics for key or indexed columns in the employee and dept tables and for the dno column in the dept table:

```
optimizedb -zk empdata -remployee -rdept -adno
```

This command performs the same operation as the previous example, but from a file:

```
optimizedb -zf flagfile
```

where flagfile contains:

```
-zk
empdata
-remployee
-rdept
-adno
```

optimizedb Example: Generate Statistics for Certain Columns and Values, in Verbose Mode

The following command does the following:

- Generates statistics for all key or indexed columns in employee, dept, and salhist.
- Processes the eno column in employee, whether or not eno is a key or indexed column.
- Generates statistics with only minimum and maximum values from the columns.
- Prints status information as each column is processed.

```
optimizedb
   -zk
   -zv
   -zx
   empdata
   -remployee
   -aeno
   -rdept
   -rsalhist;
```

optimizedb Example: Allow Unique Values from Each Column in a Table

This command allows up to 100 unique values from each column in the employee table before merging adjacent values into the same histogram cell:

```
optimizedb
   -zu100
   empdata
   -remployee;
```

printform Command—Print a Form to a File

The printform command creates a text file containing an image and description of the form and its fields.

The printform command works like the Print operation in VIFRED, which can be found on the Utilities submenu of the Forms Catalog frame.

The printform command has the following format:

```
printform dbname |vnode::dbname[/server_class] form filename [-uusername]
[-Ggroupid]
```

dbname

Identifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

form

Specifies the name of the form. You can print only one form at a time.

filename

Specifies the name of the text file to which the form is printed.

-uusername

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-Ggroupid

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

printform Example

This command prints the form employees, which is stored in the emp database, into the file emp.prf:

```
printform emp employees emp.prf
```

qbf Command—Start Query-By-Forms

The qbf command invokes Query-By-Forms (QBF), a forms-based interface for manipulating data in a database.

For a complete description of QBF, see the *Character-based Querying and Reporting Tools User Guide*.

The qbf command has the following format:

```
qbf dbname |vnode::dbname[/server_class] [-mmode] [[-t]|-f|-j|-l querytarget]
[-e] [-s] [-uusername] [-Ggroupid]
```

dbname

Identifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-m*mode*

Bypasses the Join Definition phase of QBF, putting you directly into the mode function for Query Execution, where mode is retrieve, append, update or all. If you use the -m flag, you must also specify a *querytarget*.

-t

Indicates that *querytarget* is a table. A table field format will be used to query the table. This is the default type.

-f

Indicates that *querytarget* is a QBFName. This invokes Query-By-Forms with a Visual-Forms-Editor form.

-j

Indicates that *querytarget* is a JoinDef.

-l

Allows QBF to locate the *querytarget* type. QBF searches in the order QBFName, JoinDef, table, until it finds the *querytarget* specified.

querytarget

Identifies the table, view, synonym, QBFName, or JoinDef you want to access in your query.

If you specify a query target, you must own all the tables that underlie the query target or have the proper permissions to access them. If you specify a JoinDef for the query target, you or the database administrator must own it.

Specifying *querytarget* puts you directly into the Query Execution phase. If you specify it without also specifying -m*mode*, you have the option of switching to the Join Definition phase.

The table, view, or synonym name can be qualified with a valid schema name in the format *schema*.*name*, as described in Schemas Qualifier (see page 18).

You can specify the type of *querytarget* to QBF by using the -t, -f, -j, or -l flag. If no flag is specified for *querytarget*, QBF assumes that the type is table and generates an error if it cannot find a table with that name.

-e

Invokes the command in expert mode, causing the catalogs to be displayed empty initially. Doing this allows you to enter the name of a specific object directly, rather than select it from a list.

-s

Suppresses status messages.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

qbf Examples

1. Start QBF in append mode on a custom form in the newdb database on a local node:

   ```
   qbf newdb -mappend -f myform
   ```

2. Retrieve rows from the projtasks JoinDef of the operations database on the hq remote node:

   ```
   qbf hq::operations projtasks -j -mretrieve
   ```

quel Command—Start the Line-based QUEL Terminal Monitor

The quel command invokes the line-based Terminal Monitor for QUEL.

For more information about this Terminal Monitor, see the *QUEL Reference Guide*.

The quel command has the following format:

```
quel [QUEL option flags] [line-mode flags] dbname /vnode::dbname[/server_class]
[<altin] [>altout]
```

QUEL option flags

Specify flags that can be used with the QUEL Terminal Monitor and other commands, as noted. The QUEL option flags determine the format of output or the behavior of the DBMS. You can specify a maximum of 12 QUEL option flags.

-c*N*

Sets the minimum field width for printing character columns to *N*. The default is 6.

-fkx*M.N*

Sets floating-point output column width to *M* characters (total), including N decimal places, and (if warranted), e+-xx and the decimal indicator character itself. k can be 4 or 8 to apply to f4's or f8's respectively; x can be E, F, G or N (uppercase or lowercase) to specify an output format. E indicates exponential format. F or N indicates the floating-point format. G indicates the floating-point format and guarantees decimal alignment.

If you specify F, N, or G and the number is too large for the format indicated by the flag, it is displayed in exponential format. To prevent this format overflow, M should be greater than or equal to N + 7.

If you specify F and the number is too large for the format, stars (****) are printed to represent overflow of the display.

The default display format for both f4 and f8 is n10.3, unless your computer supports the IEEE standard for floating-point numbers, in which case the display format for f4 and f8 is n11.3.

-ik*N*

Sets integer output column width to N. k can be 1, 2, or 4 for I1's, I2's, or I4's, respectively. The default for N is 6 for I1 and I2 fields, and 13 for I4 fields.

-t*N*

Sets the minimum field width for printing text columns to *N*. The default is 6.

+U|-U

Enables (+U) or disables (-U) user updating of the system catalogs and secondary indexes, and takes an exclusive lock on the database. To update system catalogs, you must have the update system tables privilege obtained through accessdb.

+Y|-Y

Enables (+Y) or disables (-Y) user updating of the system catalogs and secondary indexes, but does **not** take an exclusive lock on the database.

-uu*sername*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

-R*roleid*

Specifies a role identifier for the session, as described in Standard Flags and Parameters (see page 13).

-l

Locks the database for your exclusive use. When you specify this flag, no one else can open the database while you are in it.

If you attempt to take an exclusive lock on a database that is in use, the system informs you that the database is temporarily unavailable.

-n*M*

Sets modify mode on the index command to M. M must be one of the following storage structures: ISAM, CISAM, B-tree, CB-tree, Hash, or CHash. The default structure is ISAM.

+w|-w

Specifies wait (+w) or do not wait (-w) for the database. The default is -w. If you specify +w, there is a wait, provided that certain processes are running (sql –l, sql –U, verifydb, rollforwarddb, or sysmod) on the given database. Upon completion of those processes, the operation proceeds.

If you specify –w and the database is not available, a message is returned and execution is stopped. If you omit the w flag and the database is unavailable, an error message is returned if running in foreground (more precisely, if the standard input is from a terminal). Otherwise, the wait option is invoked.

-numeric_overflow = fail | ignore | warn

Sets error handling mode for numeric overflow, underflow and division by zero.

The fail setting causes an error message to be issued and the statement is aborted. This is the default setting. To obtain ANSI-compliant behavior, use this setting (or omit for the default).

The ignore setting causes no error message to be issued.

The warn setting causes a warning message to be issued.

-string_truncation = fail | ignore

Sets error handling mode for string truncation errors. This error occurs if you attempt to insert a string into a table column that is too short to contain the value.

The fail setting causes an error message to be issued and the statement is aborted.

The ignore setting causes no error message to be issued. The string is truncated and inserted. This is the default setting.

line-mode flags

Specifies flags that can be used with the QUEL Terminal Monitor only.

+a|-a

Sets (+a) or clears (-a) the autoclear option in the terminal monitor. The default is +a.

+d|-d

Prints (+d) or does not print (-d) the dayfile. The default is +d.

+s|-s

Prints (+s) or does not print (-s) the monitor messages, including prompts. The default is +s. If you specify -s, the dayfile is not displayed.

-v*X*

Sets the column separator to the character specified by *X*. The default is vertical bar (|).

-P*password*

Identifies the user password.

-R*role-name/role-password*

Identifies the role name and optional role password. Separate the name and password with a slash (/).

-history_recall

Invokes the terminal monitor with history recall functionality, which lets you retrieve the history of commands typed in the session, and perform other functions. For details, see the *SQL Reference Guide*.

dbname

Specifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

<altin

Specifies a file from which the Terminal Monitor reads commands. The file must contain all the Terminal Monitor commands needed to run the session. On VMS, no space is allowed between the < character and the file name.

>altout

Directs output from the Terminal Monitor to the specified file. On VMS, no space is allowed between the > character and the file name.

quel Examples

This command opens the empdata database:

```
quel empdata
```

This command opens empdata and suppresses the dayfile message:

```
quel empdata -d
```

This command opens empdata, suppresses the dayfile message and the terminal monitor prompts and messages, and reads into the workspace the contents of the batchfile:

```
quel empdata -s <batchfile
```

This command opens empdata, displays f4 columns in G format with two decimal places and I1 columns with three spaces:

```
quel empdata -f4g12.2 -I13
```

After the Terminal Monitor starts, you must use its commands to execute queries and manipulate the contents of the query buffer. For a list of Terminal Monitor commands, see Terminal Monitor Command Summary (see page 249). For details on these commands, see the Terminal Monitor chapter of the *QUEL Reference Guide*.

query Command—Invoke QBF Query Execution

The query command invokes the Query Execution phase of Query-By-Forms (QBF), a forms-based interface for manipulating data in a database. Through the Query Execution phase you can append, retrieve, or modify data. For a complete description of QBF Query Execution, see the *Character-based Querying and Reporting Tools User Guide*.

The query command has the following format:

```
query dbname |vnode::dbname[/server_class] [-mmode] [-t|-f|-j] querytarget [-e] [-
uusername] [-Ggroupid]
```

The flags and parameters have the same meaning as those for the qbf Command (see page 201), except that *querytarget* is required on the query command.

query Examples

1. Start QBF in append mode using the newdb database on a local node and a query target that is a JoinDef:

   ```
   query newdb -mappend -j staffinfo
   ```

2. Update records with the projtasks JoinDef of the operations database on the hq remote node:

   ```
   query hq::operations projtasks -j -mupdate
   ```

rbf Command—Start Report-By-Forms

The rbf command invokes Report-By-Forms (RBF), a forms-based interface that lets you build new reports or edit existing reports.

For a complete description of RBF, see the *Character-based Querying and Reporting Tools User* Guide.

The rbf command has the following format:

```
rbf dbname |vnode::dbname[/server_class] [-r|-m[style]]
report_target [-lpagewidth] [-e] [-s] [-uusername] [-Ggroupid]
```

dbname

Specifies the name of the database containing the report data, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-r

Indicates that the *report_target* is a report. If the report is found, RBF allows you to edit the report specifications. An error is returned if the named report is not found.

-m[*style*]

Indicates that the *report_target* is a table. RBF formats a default report for the specified table and then lets you edit that default report.

The optional *style* parameter specifies the style of your report. Accepted values are: wrap, tabular (same as column), block, labels, and indented. If you do not specify a style, RBF selects either tabular or block, depending on the width of your report. RBF chooses tabular if all of the columns fit on one page; otherwise RBF selects block. The default report width is 132 characters.

report_target

Specifies the name of the object that you want to access in RBF. The *report_target* can be an existing report (created in a previous RBF session), a table, view, or synonym in your database on which you want to base the report.

The table, view, or synonym name can be qualified with a valid schema name in the format *schema.name*, as described in Schema Qualifier (see page 18).

You can specify the type of *report_target* to RBF by using the –r or –m flag. If neither flag is specified, RBF looks first for a report having the specified name. If a report is not found, but a table with the same name exists, RBF sets up a default report for that table.

-l*pagewidth*

Directs RBF to use the line length specified by *pagewidth* when generating default reports. By default, RBF uses a line length of 132 characters for the label style, 80 characters for block, and 100 characters for the wrap style. All other styles, that is, tabular, indented and master-detail (a report run on a JoinDef) have no default line length. If you do not specify one for these, RBF makes the report as wide as necessary to accommodate the data.

-e

Invokes RBF in empty mode. The RBF Reports Catalog frame appears without data in its table field. This flag accelerates the process of selecting a report definition for editing, for users who are familiar with the contents of a database's reports catalog. To use this flag with a particular report definition, move the cursor to the Name column, enter the desired report name, and select the appropriate operation.

-s

Suppresses status messages.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks (**"-G*groupid*"**).

rbf Examples

1. Start RBF for the sales table in the newdb database on the hq remote node:

   ```
   rbf hq::newdb sales
   ```

2. Start RBF for the emp table owned by hr (Human Resources) in the personnel database on a local node and create a default Tabular report that is 200 characters wide:

   ```
   rbf personnel hr."perm emp" -mtabular -l200
   ```

   For information on passing the delimited identifier's surrounding quotes through your operating system, see the System Administrator Guide for the system on which your database resides.

3. Start RBF for the emp table in the personnel database on a local node and create a default report specification, letting RBF choose the Tabular or Block style, depending on the width of the report:

   ```
   rbf personnel emp -m
   ```

4. Start RBF for emp table in the personnel database on a local node and create a Labels report:

   ```
   rbf personnel emp -mlabels
   ```

5. Start RBF for the personnel database on a local node and display the emplist report specification for editing:

   ```
   rbf personnel emplist -r
   ```

   or

   ```
   rbf personnel emplist
   ```

rcpconfig Command—Control Logging and Locking System

Permission required: System administrator.

The rcpconfig utility controls the state of the logging system, and is used in system maintenance.

The rcpconfig command is called by the ingstart and ingstop commands to configure or shut down the archiver and recovery processes for your installation. Typically, you do not run the rcpconfig utility directly.

Instead of using rcpconfig, use ingstart to reconfigure the locking and logging system and ingstop to shut down the entire Ingres system, including the locking and logging system.

The rcpconfig command has the following format:

```
rcpconfig [[-init | -init_log | -init_dual [-node nodename]]| -force_init |
-force_init_log | -force_init_dual | -enable_log | -enable_dual |-disable_log |
-disable_dual | -shutdown | -imm_shutdown] [-silent] [-help]
```

-init

Initializes both transaction log files. This can be done only when the installation is offline.

In cluster installations, the node flag can also be used.

-init_log

Initializes the primary transaction log file. This can be done only when the installation is offline.

In cluster installations, the node flag can also be used.

-init_dual

Initializes the dual transaction log file. This can be done only when the installation is offline.

In cluster installations, the node flag can also be used.

-node *nodename*

Queries a specific node. A *nodename* is valid in a cluster installation only.

-force_init

Forcibly initializes both transaction log files. This can be done only when the installation is offline, but after allocating shared memory (csinstall).

-force_init_log

Forcibly initializes the primary transaction log file. This can be done only when the installation is offline.

-force_init_dual

Forcibly initializes the dual transaction log file. This can be done only when the installation is offline.

-enable_log

Enables the primary transaction log file. This can be done only when the installation is offline.

-enable_dual

Enables the dual transaction log file. This can be done only when the installation is offline.

-disable_log

Disables the primary transaction log file.

-disable_dual

Disables the dual transaction log file.

-shutdown

Gracefully shuts down the installation. It waits for any currently executing transactions to finish and cleans up the logging and locking system prior to shutdown.

-imm_shutdown

Shuts down the installation immediately, not waiting for currently executing transactions to complete. Transaction recovery will be required when the installation is restarted.

-silent

Sets the program exist status to TRUE or FALSE according to the results of the operation.

-help

Displays command syntax online.

rcpstat Command—Display Logging System Status

Permission required: Installation owner.

The rcpstat utility displays the status of the logging system. It is used for system maintenance.

The rcpstat command has the following format:

```
rcpstat [[[-exist | -format | -enable [-dual]]| -online | -transactions | -sizeok
| -csp_online [-node nodename]] | -any_csp_online] [-silent] [-help]
```

-exist

Shows whether the primary (default) or dual (dual flag) transaction log file exists.

-format

Shows whether the primary (default) or dual (dual flag) transaction log file is formatted.

-enable

Shows whether the primary (default) or dual (dual flag) transaction log file is enabled.

-dual

Indicates the dual log, when using the exist, format, or enable options.

-online

Shows whether the logging system is online or offline.

-transactions

Shows if the logging system has recoverable transactions in the transaction log file.

-sizeok

Shows whether the primary and dual transaction log files are the same size.

-csp_online

Shows whether, in a cluster installation, the Cluster server (CSP) is online on this node.

-node *nodename*

Queries a specific node, on any of the preceding flags. A *nodename* is valid in a cluster installation only.

-any_csp_online

Displays whether, in a cluster installation, the Cluster server (CSP) is online on any node.

-silent

Sets the program exist status to TRUE or FALSE according to the results of the operation.

-help

Displays command syntax online.

reconcil Command—Assist in Recovering Lost Data

The reconcil command is used with standard DBMS recovery methods to recover lost data from journals, the transaction log file, or dump areas. The cause of lost data is usually irrecoverable disk failure.

The reconcil command works only on a replicated database that is restored to a consistent state using one of the standard recovery methods, such as checkpointing, journaling, and operating system backup.

Caution! *Do not use the reconcil command as your only means of disaster recovery. It should be used only with other standard disaster recovery tools and only if the conventional methods alone are unable to recover the data.*

In the event of a system failure, it is standard procedure to restore the affected database from checkpoints; however, if journals, log files, or dump areas are lost, a gap of missing data will exist on the failed database. This gap of data may still exist on one of the database replicas. The reconcil command can recover this gap using the shadow and archive tables on the affected database, if those records still exist on these tables and reflect the data that was lost for the duration of the gap.

Note: Since the arcclean command purges records from the archive and shadow tables, you **cannot** use the reconcil command to recover lost data if you have executed arcclean on all of the replicated databases for the time of the information gap.

The reconcil command looks at each shadow table in the replicated database from a user-specified start time. If a record belongs to a CDDS that is common to the failed database, the command places an operation (insert, update, delete) for that record in the distribution queue, if the operation does not already exist in the queue. Set the collision mode of the CDDS to BenignResolution and start the necessary servers to allow the lost data to be retransmitted to the failed database.

Before you use the reconcil command, be sure you have:

- Quieted all Replicator Servers in the environment by excluding users from replicated databases.
- Ensured that all the entries in the input queue have been moved to the distribution queue.

The reconcil command has the following format:

Windows, VMS:

```
reconcil [vnode::] dbname target_db_number cdds_no|"(x,y,z,...)" |all "start_time"
[-udba_name]
```

UNIX:

```
reconcil [vnode::] dbname target_db_number cdds_no|'(x,y,z,...)' |all 'start_time'
[-udba_name]
```

target_db_number

Specifies the Ingres Replicator database number of the failed database. It must be a valid database name and Ingres Net virtual node name.

-u*dba_name*

Identifies the name of the DBA who owns the replicated database specified in *dbname*.

***cdds_no* | "(*x,y,z,...*)" |all**

Specifies the CDDSs that are to be transmitted to the failed database to bring it back in synch with its replica.

To specify the CDDS, use one of the following formats:

- cdds_no to send a single CDDS number
- '(x,y,z,...)' to send a set of CDDS numbers
- all to send all CDDS numbers

Note: If you specify more than one CDDS number, provide single quotes for UNIX and double quotes for OpenVMS and Windows.

'*start_time*'

Specifies the start time, in Ingres date and time format, used for recovering the lost data. Provide single quotes around the date and time for UNIX and double quotes for OpenVMS and Windows.

To ensure that the start time covers the duration of the information gap, be sure to specify a start time prior to the database failure. It is better to have overlapping data that can be reconciled than risk an information gap in the target database.

[*vnode*::]*dbname*

Identifies the name of the replicated database that is to provide the lost data to the failed database. The replicated database must have a Replicator Server configured to transmit the lost data to the failed database.

reconcil Example: Perform Disaster Recovery

The steps in the following scenario provide an example of how you can perform disaster recovery with the reconcil command:

1. A system failure occurring between 10:25 a.m. and 10:30 a.m. on September 20 destroys a disk on database lon::europe. A disk containing the transaction log file is also destroyed.

   As a result, there is an estimated five-minute gap in committed transactions that were in the log file after the journals were re-run.

2. The DBA recovers the database from checkpoint which brings the database lon::europe to consistency as of 10:25 a.m.

3. To recover lost data in the database from the transaction log file, the DBA selects two databases to use with the reconcil command, nyc::hq and hkg::asia, both of which are full-peer replicas of the original lon::europe database.

   The database lon::europe shares CDDS numbers 0 and 1 with nyc::hq and CDDS number 2 with hkg::asia.

4. The DBA removes databases nyc::hq and hkg::asia from user service and quiets their Replicator Servers.

5. In both databases, the DBA ensures that all the entries in the input queue have been moved to the distribution queue.

6. The DBA invokes the reconcil command on the nyc::hq and hkg::asia databases. For example, the DBA issues the following command respectively on the nyc::hq and hkg::asia UNIX machines:

   UNIX:

   ```
   reconcil nyc::hq 20 '(0,1)' '16-nov-98 10:20'
   reconcil hkg::asia -uwong 20 2 '16-nov-98 10:20'
   ```

   The target database number for both these commands is 20 (the number for lon::europe). On the nyc::hq machine, the CDDS set specified is '(0,1)', while on the hkg::asia machine, only CDDS number 2 is specified.

   Note: Since data was lost between 10:25 and 10:30, the DBA starts the reconcil command at 10:20, providing an overlap of at least five minutes to ensure the gap of missing data is recovered.

7. The DBA configures CDDSs 0, 1, and 2 with collision mode BenignResolution.

8. The DBA starts the Replicator Servers to bring the database back in synch.

relocatedb Command—Move a Location to a New Location

The relocatedb command moves the journal, dump, checkpoint, or default work location for a database to another location (when a disk fills or is swapped out, for example).

The relocatedb command can also make a copy of an entire database. Any location in the original database can be moved to a new location in the new database.

The relocatedb command has the following format:

```
relocatedb dbname[/server_class] -new_ckp_location=locationname |
-new_dump_location=locationname | -new_jnl_location=locationname |
-new_work_location=locationname | -new_database=newdbname
[-location=locationname{, locationname} -new_location=locationname
{, locationname}]
```

dbname

Specifies the name of the database whose files are to be moved, and if required, the *server_class*, as described in Standard Flags and Parameters (see page 13).

-new_ckp_location=*locationname*

Specifies the name of the new checkpoint location. The *location* must be defined with checkpoint usage. Checkpoints should not be run when relocating a checkpoint location.

-new_dump_location=*locationname*

Specifies the name of the new dump location. The *location* must be defined with dump usage. Checkpoints should not be run when relocating a dump location.

-new_jnl_location=*locationname*

Specifies the name of the new journal location. The *location* must be defined with journal usage. Journaling and checkpoints should not be run when relocating a journal location.

-new_work_location=*locationname*

Specifies the name of the new work location. The database must have been previously extended to this *location* for work usage. An exclusive lock is required during the relocation operation.

-new_database=*newdbname*

Specifies the name of the new database to be created.

This option creates all the directories and copies all the files from the old database to the new database.

An exclusive lock on the original database is required while the original database is copied to the new database.

Note: Files for data recovery (*.ckp, *.jnl, *.dmp) are applicable to the original database only and are not valid for the new database. Following a relocatedb -new_database=*newdbname*, the new database should have a checkpoint taken with the appropriate journaling flags set according to its usage.

-location=*locationname*(, *locationname*)

Specifies a list of locations. All locations in the list must be valid locations for the database. The physical files in the data areas are copied to the same location for the new database.

When new_location is specified, the physical files in each data area in the location list is copied to the corresponding data area in the new location list.

This option is valid only for database relocation (-new_database=*newdbname*).

-new_location=*locationname*(, *locationname*)

Specifies a list of new locations. All locations in the list must be defined for the installation, and the usage must be compatible with the usage of the corresponding area in the location list. Each location in the location list must be mapped to a distinct location in the new location list.

This option is valid only for database relocation (-new_database**=***newdbname*).

relocatedb Example: Relocate Checkpoint Files

The following command relocates the checkpoint location for the empdata database to newckp:

```
relocatedb empdata -new_ckp_location=newckp
```

This command example does the following:

- Performs an update to the iidbdb: update iidatabase.ckpdev='newckp' where name='empdata'. (The update can be verified by connecting to the iidbdb database and doing a select from iidatabase where name='empdata'.)
- Updates the checkpoint location in the configuration file. (This can be verified by examining the output of the infodb empdata command.)
- Copies checkpoint files from the old checkpoint location to the new checkpoint location.
- Deletes checkpoint files from the old checkpoint location.

relocatedb Example: Relocate Journal Files

The following command relocates the journal location for the empdata database to newjnl:

```
relocatedb empdata -new_jnl_location=newjnl
```

This command example does the following:

- Performs an update to the iidbdb: update iidatabase.jnldev='newjnl' where name='empdata'. (This can be verified by connecting to the iidbdb database and doing a select from iidatabase where name='empdata'.)
- Updates the journal location in the configuration file. (This can be verified by examining the output of the infodb empdata command.)
- Copies journal files from the old journal location to the new journal location.
- Deletes journal files from the old journal location.

relocatedb Example: Relocate Dump Files

The following command relocates the dump location for the empdata database to newdump:

```
relocatedb empdata -new_dump_location=newdump
```

This command example does the following:

- Performs an update to the iidbdb: update iidatabase.dmpdev='newdump' where name='empdata'. (This can be verified by connecting to the iidbdb database and doing a select from iidatabase where name='empdata'.)
- Updates the dump location in the configuration file. (This can be verified by examining the output of the infodb empdata command.)
- Copies dump files from the old dump location to the new dump location.
- Deletes dump files from the old dump location.

relocatedb Example: Relocate the Work Area

The following command relocates the work location for the empdata database to newwork:

```
relocatedb empdata -new_work_location=newwork
```

The command above does the following:

- Performs an update to the iidbdb: update iidatabase.sortdev='newwork' where name='empdata'. (This can be verified by connecting to the iidbdb database and doing a select from iidatabase where name='empdata'.)
- Updates the default work location in the configuration file. (This can be verified by examining the output of the infodb empdata command.)
- No files are copied for work locations.

relocatedb Example: Copy Database to a New Database

The following command copies the empdata database to a new database called empdev:

```
relocatedb empdata -new_database=empdev
```

This command example does the following:

- Inserts a record in the iidatabase table for this database.
- Inserts a record in the iiextend table for all locations to which the new database is extended.
- Creates all the directories and copies all the files from the old database to the new database.
- Builds a database configuration file for the new database.

After the new database is created, you should be able to connect to it and access all data.

repcat Command—Create and Load Replicator Catalogs

The repcat command creates and populates the Ingres Replicator catalog tables and creates Ingres Replicator database events. You must run repcat before starting the Replicator Manager for the first time.

The repcat command must be run before you can move the Ingres Replicator configuration from a configuration database to the other participating databases. The repcat command must be run on every Replicator database containing Full Peer or Protected Read-only CDDSs. You do not need to run repcat on databases with only Unprotected Read-only CDDSs.

Note: You can run repcat from a single location by specifying the virtual node name (*vnode*).

The repcat command has the following format:

```
repcat [+w|-w] [-udba_name] [vnode::]dbname
```

+w|-w

Indicates whether to wait for an exclusive lock on the database.

Default: -w (do not wait)

-u*dba_name*

Specifies the effective user for the session. You must run repcat as the owner of the database.

dbname

Identifies the name of the database, and if required, the *vnode*, as described in Standard Flags and Parameters (see page 13).

repcat Examples

This command creates and loads Replicator catalogs for the europe database:

```
repcat europe
```

This command creates and loads Replicator catalogs for database hq on vnode nyc, with the Replicator database administrator as user:

```
repcat -urep_dba nyc::hq
```

repcfg Command—Configure Replicator

The repcfg command lets you configure Replicator installations from the command line (instead of using Replicator Manager or Visual DBA).

The repcfg command has the following format:

```
repcfg dbname obj_type action [-uusername] [-q] object {object} [-help]
```

dbname

Specifies the name of the database.

obj_type

Specifies the object type: cdds or table. Both arguments can be abbreviated to the initial character and are case-sensitive.

action

Specifies the action to be performed:

activate

Activates the specified object.

deactivate

Deactivates the specified object.

createkeys

Creates replicated transaction keys for every row in the base table and populates the shadow table. It can also populate the input queue when used with the -q flag. You must use the option if you install Replicator on an existing database that contains data.

These arguments can be abbreviated to the initial character and are case-sensitive.

-q

Populates the shadow table and the input queue. This option is used only when the action parameter is createkeys. Use this option if your databases are not synchronized; the altered rows are placed in the input queue for reconciliation or distribution. This option also must be used if the table is horizontally partitioned.

-u*username*

Specifies the effective user for the session.

object

Specifies the object to configure. The object can be a CDDS number or table number.

Note: No more than 100 objects can be specified. This limit can be raised by modifying the utexe.def file. For more information, see the *Database Administrator Guide*.

repcfg Examples

This command activates CDDS 0 in the repdb database:

```
repcfg repdb cdds activate 0
```

This command creates replication keys for tables 3 and 4 in the europe database and populates the input queue:

```
repcfg europe table createkeys -q 3 4
```

repdbcfg Command—Configure Multiple Mobile Databases

The repdbcfg command allows you to configure multiple Ingres mobile databases simultaneously from the operating system prompt of the host machine to which Ingres Replicator on mobile Replicator will connect.

Note: In Full Peer situations, the paths created by repdbcfg will not be sufficient to permit all changes from each mobile database to be replicated to all other mobile databases. For more information, see the *Replicator User Guide*.

The repdbcfg command has the following format:

```
repdbcfg [vnode::]dbname filename [-udba_name]
```

vnode*::*dbname

Specifies the name of the database to be configured.

filename

Specifies the name of the input file.

The input file should have the following format:

```
user_name [db_no] [dbname] [cdds_no] [target_type]
```

where:

user_name

Specifies the user name of the owner of the mobile database. The *user_name* must be unique and 32 bytes or less.

db_no

Specifies the number of the mobile database.

The optional db_no must be a number in the range of 1-32,767 that has not already been used in your Replicator configuration (dd_databases table). If db_no is not provided on the first line in the file, repdbcfg uses 101 as a default or the next higher number that does not exist in dd_databases. If db_no is not specified on subsequent lines, the previous value incremented by one is used.

dbname

Identifies the name of the mobile database. The optional *dbname* should be a unique database name up to 32 bytes long. If it is not provided, the *user_name* is used as the default database name. The node name for all mobile databases is "mobile."

cdds_no

Specifies the number of the CDDS for the mobile database.

The optional cdds_no should be a number in the range of 1-32,767. It should already be defined in the dd_cdds table through the CDDS Detail screen. If it is not provided on the first line, repdbcfg uses 50 as the default. If it is not provided on subsequent lines, the previous value is used.

target_type

Specifies the CDDS target type of the mobile database. Valid values are:

FP – Full Peer
PR – Protected Read-only
UR – Unprotected Read-only

Default: If *target_type* is not provided on the first line, the default is FP. If it is not provided on subsequent lines, the default is the previous value.

For each *target_type*, repdbcfg creates a data propagation path, under the given or default CDDS, from the local database to the mobile database. If the *target_type* is FP, repdbcfg creates an additional path from the mobile database to the local database.

repdbcfg Examples

1. In this example, repdbcfg is invoked against an input file containing 26 user names:

```
albert
barbara
charlie
...
zoe
```

The repdbcfg utility defines 26 new full peer mobile databases, numbered 101 through 126. Connection names are in the form "mobile::charlie." Data propagation paths are added for CDDS 50 to and from the local database and each new mobile database. If the local database number is 5, the first four paths are 5-5-101, 101-101-5, 5-5-102, and 102-102-5.

2. In this example, the repdbcfg uses the input file:

```
albert 201 albert 3 FP
barbara
charlie 301 charlie 4 PR
daniel
```

The repdbcfg utility defines four new mobile databases numbered 201, 202, 301, and 302. The first two are full peer and each has two data propagation paths for CDDS 3. The other two databases are protected read-only and each has one propagation path for CDDS 4.

repinst Command—Create or Remove Replicator Servers and Windows Services

Valid on Windows.

The repinst command creates or removes Ingres Replicator servers as Windows services.

The repinst command has the following formats:

```
repinst num_servers
```

```
repinst remove
```

num_servers

Specifies the number of Replicator Servers to install. Servers are numbered sequentially starting from 1. If some services have already been created, repinst only creates new services beyond the existing ones, but up to *num_servers*.

remove

Removes all services.

repinst Examples

This command creates three services:

```
repinst 3
```

This command creates two more services:

```
repinst 5
```

This command removes all five services:

```
repinst remove
```

repmgr Command—Start Replicator Manager

The repmgr command invokes the Replicator Manager, which the distributed DBA uses to administer Ingres Replicator.

To run Replicator Manager, you must have the correct system privileges and use the correct DBA name. An Ingres user with security privilege can impersonate the DBA and modify the information in Replicator Manager.

The repmgr command has the following format:

```
repmgr [-udba_name] [vnode::]dbname
```

-u*dba_name*

Specifies the effective user for the session. You must run repmgr as the owner of the database.

[*vnode*::]*dbname*

Specifies the database to connect to. By specifying the *vnode* of a remote database in Replicator Manager, the DBA can administer the Ingres Replicator network from the local machine.

repmgr Example

This command, executed from a remote San Francisco computer, allows Replicator Manager to be run in client-server mode from the San Francisco computer to the hq database on the nyc node, and assumes that the San Francisco DBA is impersonating the New York DBA (nyc_dba):

```
repmgr -unyc_dba nyc::hq
```

repmod Command—Modify Replicator System Tables Storage Structure

The repmod command modifies Ingres Replicator system tables in a replicated database to predetermined storage structures. The tables are modified to the most appropriate storage structure for accelerating query processing. You must run repmod on the whole database.

The repmod command has the following format:

```
repmod [vnode::] dbname [- udba_name] [+w|-w]
```

[*vnode*::] *dbname*

Identifies the name of the database whose system tables are to be modified.

-u*dba_name*

Specifies the effective user for the session. You must run repmod as the owner of the database.

+w|-w

Directs repmod to wait (+w) or not wait (-w) until the database is free before executing. Repmod requires exclusive access to the database.

On VMS, this flag is not valid in batch mode.

report Command—Run a Report on a Table

The report command runs a report for a table in the database. The command creates either a default report or a report set up by the rbf or sreport command.

For a complete description of Ingres reporting tools, see the *Character-based Querying and Reporting Tools User Guide*.

The report command has the following format:

```
report dbname |vnode::dbname[/server_class] [-r|-m [style]] report_target
(variable=value {,variable=value})] [-foutputfile] [-oprinter] [-ncopies] [-5]
[-6] [+b|-b] [-d] [-h] [-lpagewidth] [-qmxquer] [+t|-t] [-vpagelength] [-wmxwrap]
[-ifilename] [-s] [-uusername] [-Ggroupid] [-numeric_overflow=fail|ignore|warn]
```

dbname

Identifies the name of the database containing the report data, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-r

Indicates that a report is specified as the report_target. If the specified report is not found, an error message is returned.

-m[*style*]

Indicates that a table is specified as the report_target. This instructs report to format a default report for the specified table.

The optional style specifies the style of your report. Accepted values are wrap, tabular (same as column), block, labels, and indented. If you do not specify a style, report selects either tabular or block, depending on the width of your report. Tabular is used if all of the columns fit on one page; otherwise, block is selected. The default report width is 132 characters.

report_target

Specifies the name of the object on which you wish to run the report. The *report_target* can be:

- An existing report, created using RBF or sreport.
- A table, view, or synonym in your database on which you want a default report formatted.

The table, view, or synonym name can be qualified with a valid schema name in the format schema**.**name, as described in Schema Qualifier (see page 18).

You can specify the type of report_target by using the –r or –m flag. If neither flag is specified, the report command looks first for a report having the specified name. If a report is not found, but a table with the same name exists, report sets up a default report for that table.

variable=value

Specifies the name of a parameter (*variable*) used in the report, and the *value* that is replaced for every occurrence of the corresponding *variable* name in the report specifications. If you want to specify a string or a date, *value* must be quoted. You can separate variable and value combinations using blanks, tabs, or commas.

-f*outputfile*

Directs the formatted report to the outputfile. If this option is not specified, the report is written to the standard output file (normally your terminal), or, in the case of a report specified by the Report Writer, to the file designated by the .output command in the report specification file.

-o*printer*

Sends the report to the specified printer.

To set a default printer, define ING_PRINT. If you require special print options, specify the options in ING_PRINT, and specify the –o flag with no argument.

-n*copies*

Specifies the number of copies of the report to print.

-5

Forces version 5 compatibility mode, as follows:

The +t option is the default for aggregates.

All arithmetic is floating-point, unless all values in the computation are integers.

By default, the month portion of the current_date() function is displayed in uppercase letters.

-6

(SQL reports only) Eliminates duplicate rows from reports whose specification contains .data, .table, .view, or .sort statements.

-b|+b

Forces (+b) or suppresses (-b) form feed at the end of each page. The flag overrides formfeed or .noformfeed commands in the report specification file.

-d

Directs report to continue running the specified report if the .setup or .cleanup statements generate DBMS errors.

-h

Provides a null set of data for a report that retrieves no rows. All .header and .footer sections are executed. The detail section is suppressed. This feature allows you to include the following .if statement in the report footer to indicate that no rows were found:

```
if count(column) = 0 .then
  .print
  “No data matched the
   query specifications.”
.endif
```

-l*pagewidth*

Sets the maximum output line size to *pagewidth* characters. By default, if output is to a file, the maximum output line size is 132 characters; otherwise, the default maximum line size is the width of the terminal.

-q*mxquery*

Sets the maximum length of the query after all substitutions for runtime parameters have been made to *mxquery* characters. By default, the maximum query size is 2048 characters. This option is needed for long queries only.

-t|+t

Causes aggregates and breaks to use using underlying values (-t) or rounded values (+t) for any floating-point column whose format has been specified in a .format command as numeric F or template. If +t is specified, each value in the column is rounded to the precision given by its format, and breaks for date columns that use a date template occur over the actual value appearing for the dates.

Default: –t.

-v*pagelength*

Sets the number of lines for each page of output. The *pagelength* must be a positive integer. This flag overrides any .pagelength command in the report specification file.

Default: 61 lines per page if the report is written to a file; 23 lines per page if written to a terminal.

-w*mxwrap*

Specifies the maximum number of lines to wrap (*mxwrap)* with one of the column C formats, or the maximum number of lines that can be used in any block.

Default: 300. This maximum is provided as a protection against misspecified columns and is rarely needed.

-i*filename*

Reads a report specification from the specified file outside of the database, and runs the report. Using this flag eliminates the need to use the sreport command to place the report source file in the database for processing.

When using this flag, you must omit *report_target* and the –r|-m flag.

-s

Suppresses status messages.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks (**"-G***groupid*").

-numeric_overflow=fail | ignore | warn

Sets error-handling mode for numeric overflow.

fail

(Default) Causes an error message to be issued and the statement is aborted. To obtain ANSI-compliant behavior, use this setting (or omit for the default).

ignore

Causes no error message to be issued.

warn

Causes a warning message to be issued.

report Examples

1. Run a default report based on the vendor table in the purchasing database, using a default format and redefining the page width, and send the output to the named printer:

   ```
   report purchasing vendor -m -l80 -olaser2
   ```

2. Run the report contained in the source file, po_rep.rw, against the purchasing database and store the results in the po_sum.out file:

   ```
   report purchasing -ipo_rep.rw -fpo_sum.out
   ```

3. Run a default report in column format on the clients table owned by mktgmgr in the sales database and eliminate duplicate rows:

   ```
   report sales mktgmgr.clients -mcolumn -6
   ```

4. Run the report named recpay against the accounting database and pass in the value of the variable, title:

   ```
   report accounting recpay
      (title = 'AccountsReceivable')
   ```

repstat Command—Display Replicator Transaction Statistics

The repstat command provides statistics about Ingres Replicator transactions, which include the Mutex address, queue size in entries, the start and end of queue at entry number, and the number of entries currently in the queue.

The repstat command has the following format:

```
repstat
```

rmcmdgen Command—Generate VDBA Remote Command Catalogs

Permission required: Installation owner.

The rmcmdgen utility generates the Visual DBA remote command catalogs. It generates the objects in the iidbdb database that are needed by rmcmd.

The rmcmdgen utility is typically invoked by the installation program. It can be started only by the user that owns the installation, and it has no command line parameters.

The rmcmdgen command has the following format:

```
rmcmdgen
```

rmcmdrmv Command—Remove VDBA Remote Command Catalogs

Permission required: Installation owner.

The emcmdrmv utility removes the Visual DBA remote command catalogs. It removes (from the iidbdb database) the objects (tables, views, procedures, dbevents) that are needed by rmcmd.

The rmcmdrmv utility is typically used when upgrading an installation. It can be executed only by the user that owns the installation, and it has no command line parameters.

The rmcmdrmv command has the following format:

```
rmcmdrmv
```

rmcmdstp Command—Stop the Remote Command Process

Permission required: Installation owner.

The recmdstp utility stops rmcmd, the remote command process. Typically, rmcmd is stopped by the ingstop process.

The rmcmdstp utility can only be executed by the user that owns the installation, and has no command line parameters.

The rmcmdstp command has the following format:

```
rmcmdstp
```

rollforwarddb Command—Recover a Database

Permission required: DBA or a system administrator running rollforwarddb with the –u flag. On VMS, if using this command against a database in a group level installation, you must have the VMS CMKRNL privilege.

The rollforwarddb command recovers a database or table from the last checkpoint and the current journal and dump files. When executing table level recovery, you can optionally move the table to a new location.

If the target checkpoint was performed online (while the database was in use), then rollforwarddb does the following:

1. Restores the database from the checkpoint location to the database location
2. Applies the log records in the dump location to the database, which returns the database to its state when the checkpoint began
3. Applies the journal records to the database

If the target checkpoint was executed offline, then the second step is omitted.

By default, rollfrowarddb sequentially restores data locations one at a time. A database with more than one data location can be restored in parallel.

For detailed procedures on performing backup and recovery of the database, see the *Database Administrator Guide*.

The rollforwarddb command has the following format:

```
rollforwarddb dbname[/server_class] [+c|-c] [+j|-j] [#m[n]]
[-mdevice{, device}] [-bdd-mmm-yyyy:hh:mm:ss[.cc]] [-edd-mmm-yyyy:hh:mm:ss[.cc]]
[#c[n]] [+w|-w] [-v] [#f] [-uusername] [-statistics] [-incremental] [-norollback]
[-table=tablename{, tablename} [-nosecondary_index]
[-ignore] [-on_error_continue] [-on_error_prompt]
[-relocate -location=locationname {, locationname}
-new_location=locationname{, locationname}]]
[-dmf_cache_size= x] [-dmf_cache_size_4k|8k|16k|32k|64k= x] [-help]
```

dbname

Identifies the database (one database name only) to be recovered, and if required, the *server_class*, as described in Standard Flags and Parameters (see page 13).

+c|-c

Recovers (+c) or do not recovers (-c) the database from the checkpoint file. The default is +c.

+j|-j

Recovers (+j or do not recovers (-j) the database from the journal. The default is +j.

-m*device* {, *device*}

Recovers the checkpoint from the specified tape device. If a list of tape devices is supplied, parallel recovery will be used for a multi-location database.

If the database was checkpointed to a tape, you can use the –m flag to restore the database from the tape.

VMS: Before executing rollforwarddb from a tape device, the tape must be inserted into the tape drive.

-b*dd-mmm-yyyy*[:*hh*:*mm*:*ss*[.*cc*]]

Recovers transactions that were completed after the specified date and time only. Fractional seconds are optional and assumed to be ".00" if not specified.

-e*dd-mmm-yyyy*[:*hh*:*mm*:*ss*[.*cc*]]

Recovers transactions that were completed before the specified date and time only. Fractional seconds are optional and assumed to be ".00" if not specified.

Note: The -e and -b flags are fully supported when used against an entire database.

Caution! *Using the -b or -e options with the -table flag will result in the table being logically inconsistent. Using these parameters to skip recovery of a segment of the journal file is not supported.*

-norollback

Bypasses the rollback phase of rollforward and leaves the database in the exact state described by the journal files. If used with the -e flag, then the database is left in the state described by the journal files up to the time specified. Any incomplete transactions are left incomplete. If transactions are left incomplete, the database will be in an inconsistent state after the rollforwarddb.

Note: The default is -rollback, which does not need to be explicitly specified.

#c[*n*]

Recovers from an older checkpoint. The checkpoint number n must be a valid checkpoint number (as shown by the infodb command). This flag can be used to recover the database when the current checkpoint is unfinished. If n is omitted, the most recent usable finished checkpoint is used for the recovery.

UNIX: In bash shell, place this option in quotes; otherwise characters after the # are treated as a comment. For example:

```
rollforwarddb empdata "#c1"
```

For a discussion of limitations and cautions when recovering from older checkpoints, see the *Database Administrator Guide*.

+w|-w

Waits (+w) or does not wait (-w) for the database to be free (not in use). The default is -w.

VMS: The +w|-w flag directs rollforwarddb to wait (+w) or not wait (**-**w) for the database to be free before recovering the database. Since rollforwarddb requires the database to be locked, this flag allows you to decide whether to wait for the database to be free if it is in use. If you specify +w, rollforwarddb will wait as long as necessary for the database to become free for locking and recovery. If you specify –w, an error is returned if the database is busy. The default is –w.

This flag can be used only in interactive sessions and not in batch mode.

#m[*n*]

Recovers *n* locations at a time from disk, for a multi-location database.

UNIX: In bash shell, place this option in quotes; otherwise characters after the # are treated as a comment.

-v

Recovers the database from the journal in verbose mode, which provides diagnostic information about all operations executed during the recovery process.

#f

Forces journaling enablement, if rollforwarddb with journaling is attempted on a database that has journaling disabled.

UNIX: In bash shell, place this option in quotes; otherwise characters after the # are treated as a comment.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-statistics

Prints statistics about the rollforwarddb operation.

-table=*tablename*{, *tablename*}

Specifies a list of tables to be recovered from the target checkpoint. If multiple tables are specified, no space is allowed between the tables listed. Table recovery is not allowed for views, system catalogs, or Enterprise Access tables.

If recovering a base table, blob columns (long byte and long varchar columns) will be recovered, and secondary indexes will be recovered, unless –nosecondary_index is specified.

-nosecondary_index

Inhibits automatic recovery of secondary indexes.

Note: All secondary indexes will be marked inconsistent. The base table cannot be accessed until the secondary indexes are rebuilt or dropped.

This option is invalid for database level recovery.

-ignore

Ignores any errors that occur during the processing of journal records, and applies subsequent records for the table. The table will be marked inconsistent at the end of rollforwarddb. The database will also be marked inconsistent to bring to your attention the errors that occurred during rollforwarddb. Choose this action when the table cannot be rebuilt from another source and you want to try to recover as much data as possible.

-on_error_continue

Continues processing journal records if an error occurs, but does not apply subsequent records for the table. The table is removed from the table list and processing continues. The table or index will be marked inconsistent at the end of rollforwarddb. The database will also be marked inconsistent to bring to your attention the errors that occurred during rollforwarddb. Choose this action for secondary indexes (which can be rebuilt) or if the table can be rebuilt from another source.

If this option is not specified and an error is encountered, all tables being recovered are marked inconsistent and rollforwarddb terminates.

Note: This option does not force continuation of an invalid rollforwarddb command. The rollforwarddb process is terminated immediately if an invalid table—for example, a view, system catalog, Enterprise Access table, nonexistent table, or a table for which recovery is disallowed—is specified.

This option is invalid for database level recovery.

-on_error_prompt

Prompts "Error during recovery of table (*tablename*, *tableowner*)" if an error occurs when applying dump or journal records. The on_error_prompt option provides the ability to handle errors for various tables differently.

You can respond with one of these actions:

CONTINUE_IGNORE_TABLE

Continues journal processing, but ignores subsequent journal records for this table or index. This action is equivalent to the on_error_continue flag.

CONTINUE_IGNORE_ERROR

Continues journal processing and applies subsequent records for this table. This action is equivalent to the –ignore flag.

ROLLBACK_TRANSACTIONS

(Default) Stops processing the journals and rolls back open transactions. The database is left at a consistent state, but not all updates have been reapplied and the database is inconsistent with the journals.

-incremental

Indicates an incremental rollforwarddb.

When -incremental +c -j is specified, the checkpoint is restored and rollforwarddb marks the database INCONSISTENT with inconsistency code INCR_RFP. While the database is inconsistent (INCR_RFP), you can still connect and perform read only operations.

When -incremental -c +j is specified:

- Rollfowarddb applies all new journal files that have been moved into the journal directory.
- If -norollback has also been specified, any open transaction context is written at the end of the last journal file processed.
- Before processing new journals, open transaction context is restored from the previous incremental roll forward.
- If -rollback is specified, after journal processing has finished, any open transactions are rolled back. The database will be marked consistent and updatable. This option should be specified when you have finished the incremental rollforwarddb.

Note: Incremental rollforwarddb requires that all journals since the last checkpoint be present. For example, if you apply a batch of journal files, and then delete the previous batch of journal files, **rollforwarddb -incremental -rollback** may fail.

-relocate

Indicates that a table is to be relocated to a new location during recovery. When using this option, –location and –new_location must also be specified.

This option is invalid for database-level recovery.

-location=*locationname{, locationname}*

Specifies a data location or list of locations (*locationname*).

When –relocate and –new_location are also specified, the data in each area in the location list is moved to the corresponding area in the new location list. Only tables being recovered are relocated.

This option is invalid for database level recovery.

-new_location=*locationname*{, *locationname*}

Specifies a new data location or list of new data locations (*locationname*).

When -relocate and -location are also specified, the data in each area in the location list is moved to the corresponding area in the new location list. Only tables being recovered are relocated.

When this option is specified, –relocate and -location must also be specified, and the number of locations in the location list must equal the number of locations in the new location list. (The number of location names associated with a table cannot be changed using rollforwarddb.)

This option is invalid for database level recovery.

[-dmf_cache_size= *x*] [-dmf_cache_size_4k|8k|16k|32k|64k= *x*]

Specifies the size of the local cache that rollforwarddb allocates, in number of buffers.

Default values are:

-dmf_cache_size=256, which indicates 256 2 KB buffers.

-dmf_cache_size_*x*k=200, where *x* is the buffer size indicated in the keyword. For example, -dmf_cache_size_64k=200 indicates 200 64 KB buffers.

If you specify 0 for the 4k, 8k, 16k, 32k, or 64k buffers, 256 buffers are allocated.

-help

Displays command syntax online.

rollforwarddb Examples

1. The following command recovers the empdata database from the target checkpoint and journal, and provides diagnostic information about all operations executed during the recovery process.

   Note: Both the journal and the checkpoint must be online before executing the command.

   ```
   rollforwarddb empdata -v
   ```

2. This command recovers tables emp and emphist from the empdata database:

   ```
   rollforwarddb empdata -table=emp,emphist
   ```

3. This command recovers tables emp and emphist from the empdata database without recovering the indexes:

   ```
   rollforwarddb empdata -table=emp,emphist -nosecondary_index
   ```

   Note: The indexes on tables emp and emphist will have to be rebuilt or dropped before the tables can be accessed.

4. This command recovers table emp in the empdata database and relocates it from location emploc to the new location newemploc:

   ```
   rollforwarddb empdata -table=emp,emphist -relocate -location=emploc -
   new_location=newemploc
   ```

   UNIX:

   ```
   rollforwarddb empdata +c +j -m/dev/rmt0
   ```

   VMS:

   ```
   rollforwarddb empdata +c +j -mMTA0:
   ```

rpserver Command—Start Replicator Server

The rpserver command starts a Replicator Server.

The command takes a single parameter. All other parameters are read in from the runrepl.opt file, which must be present in the corresponding server directory.

The rpserver command has the following format:

```
rpserver n
```

n

Specifies the number of the server you want to start.

rsstatd Command—Display Replicator Server Statistics

The rsstatd command displays Ingres Replicator Server statistics.

The rsstatd command has the following format:

```
rsstatd
```

sql Command—Start the Line-based SQL Terminal Monitor

The sql command invokes the line-based Terminal Monitor for SQL. For procedures on using this Terminal Monitor, see the *SQL Reference Guide.*

The sql command has the following format:

```
sql [SQL option flags] [line-mode flags] dbname |vnode::dbname[/server_class]
[<altin] [>altout]
```

SQL option flags

Specifies flags that can be used with the line-based Terminal Monitor and other commands where noted. The SQL option flags determine the format of output or the behavior of the DBMS. You can specify a maximum of twelve SQL option flags. The flags are as follows:

-c*N*

Sets the minimum field width for printing character columns to *N*. The default is 6.

-fkx*M.N*

Sets floating-point output column width to *M* characters (total), including *N* decimal places, and (if warranted) *e+-xx* and the decimal indicator character itself.

k can be 4 or 8 to apply to f4's or f8's respectively.

x can be E, F, G or N (uppercase or lowercase) to specify an output format. E indicates exponential format. F indicates floating-point format. G indicates the floating-point format and guarantees decimal alignment. N indicates floating-point format, decimal alignment, and right-justification.

If you specify N or G and the number is too large for the format indicated by the flag, it is displayed in exponential format. To prevent this format overflow, *M* should be greater than or equal to *N* + 7.

The default display format for both f4 and f8 is n10.3, unless your computer supports the IEEE standard for floating-point numbers, in which case the display format for f4 and f8 is n11.3.

-ik*N*

Sets integer output column width to N. k can be 1, 2, or 4 for i1's, i2's, or i4's, respectively. The default for N is 6 for i1 and i2 fields, and 13 for i4 fields.

-t*N*

Sets the minimum field width for printing text columns to *N*. The default is 6.

+U|-U

Enables (+U) or disables (-U) user updating of the system catalogs and secondary indexes, and takes an exclusive lock on the database. To update system catalogs, you must have the update system tables privilege obtained through accessdb.

On VMS, enclose this flag in double quotation marks ("+U" or "-U").

+Y|-Y

Enables (+Y) or disables (-Y) user updating of the system catalogs and secondary indexes, but does **not** take an exclusive lock on the database.

On VMS, enclose this flag in double quotation marks ("+Y" or "-Y").

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this flag in double quotation marks ("-G*groupid*").

-R*roleid*

Specifies a role identifier for the session, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this flag in double quotation marks ("-R*roleid*")

-l

Locks the database for your exclusive use. When you specify this flag, no one else can open the database while you are in it. If you attempt to take an exclusive lock on a database that is in use, the system informs you that the database is temporarily unavailable.

-n*M*

Sets modify mode on the index command to *M*. *M* must be one of the following storage structures: ISAM, CISAM, B-tree, CB-tree, Hash, or CHash. The default is ISAM.

+w|-w

Indicates to wait (+w) or not wait (-w) for the database. The default is -w. If you specify +w, you must wait if certain processes are running (sql –l, sql –U, verifydb, rollforwarddb, or sysmod) on the given database, before the operation proceeds.

If you specify –w and the database is not available, a message is returned and execution is stopped. If you omit the w flag and the database is unavailable, an error message is returned if running in foreground (more precisely, if the standard input is from a terminal). Otherwise, the wait option is invoked.

On VMS, this flag is not valid in batch mode.

-numeric_overflow = fail | ignore | warn

Sets error handling mode for numeric overflow, underflow and division by zero.

The fail setting causes an error message to be issued and the statement is aborted. This is the default setting. To obtain ANSI-compliant behavior, use this setting (or omit for the default).

The ignore setting causes no error message to be issued.

The warn setting causes a warning message to be issued.

-string_truncation = fail | ignore

Sets error handling mode for string truncation errors. This error occurs if you attempt to insert a string into a table column that is too short to contain the value.

The fail setting causes an error message to be issued and the statement is aborted.

The ignore setting causes no error message to be issued. The string is truncated and inserted. This is the default setting.

line-mode flags

Specify flags that can be used with the line-based Terminal Monitor only. The flags are as follows:

+a|-a

Sets (+a) or clears (-a) the autoclear option in the terminal monitor. The default is +a.

+d|-d

Prints (+d) or does not print (-d) the dayfile. The default is +d.

+s|-s

Prints (+s) or does not print (-s) the monitor messages, including prompts. The default is +s. If you specify -s, the dayfile is not displayed.

-v*X*

Sets the column separator to the character specified by *X*. The default is vertical bar (|).

-P*password*

Specifies the user password.

-R*role-name/role-password*

Identifies the role name and optional role password. Separate the name and password with a slash (/).

-history_recall

Invokes the terminal monitor with history recall functionality, which lets you retrieve the history of commands typed in the session, and perform other functions. For details, see the *SQL Reference Guide*.

dbname

Specifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

<altin

Specifies a file from which the Terminal Monitor reads commands. The file must contain all terminal monitor commands needed to run the session. On VMS, no space is allowed between the < character and the file name.

>altout

Directs output from the Terminal Monitor to the specified file. If you specify this parameter, you will not see any output. On VMS, no space is allowed between the > character and the file name.

sql Examples

This command opens the line-based Terminal Monitor for SQL for working in the empdata database:

```
sql empdata
```

This command opens empdata and does not print the dayfile:

```
sql empdata -d
```

This command opens empdata, suppressing the dayfile message and the terminal monitor prompts and messages, and reads into the workspace the contents of the batchfile:

```
sql empdata -s <batchfile
```

This command opens empdata, displays f4 columns in G format with two decimal places and i1 columns with three spaces:

```
sql empdata -f4g12.2 -I13
```

Terminal Monitor Command Summary

After the Terminal Monitor starts, you must use its commands to execute queries and manipulate the contents of the query buffer. Terminal Monitor commands are summarized here. For details, see the Terminal Monitor chapter of the *SQL Reference Guide*.

\g *or* \go

Processes the current query.

\q *or* \quit

Exits the Terminal Monitor.

\r *or* \reset

Erases the query buffer.

\p *or* \print

Prints the current query.

\e *or* \ed *or* \edit *or* \editor [*filename*]

Invokes a text editor.

\time *or* \date

Prints the current time and date.

\a *or* \append

Appends to the query buffer.

\s *or* \sh *or* \shell

Escapes to the operating system.

\cd *or* \chdir *dir_name*

Changes the working directory of the monitor to the named directory.

\i *or* \include *or* \read *filename*

Reads the named file into the query buffer.

\w *or* \write *filename*

Writes the contents of the query buffer to the named file.

\script [*filename*]

Toggles between logging and not logging the session to a file.

\[no]suppress

Suppresses or does not suppress the printing of the results returned from the query.

\[no]bell

Includes or does not include a bell with the continue or go prompt.

\[no]continue

Continues or does not continue statement processing on error.

sreport Command—Store Report Definition in a Database

The sreport command writes a Report Writer report definition into the specified database. You can also use this command with the copyrep command to copy a report from one database to another, or from one owner to another.

The sreport command reads in a file of Report-Writer source code formatting statements from a base text file, as well as from any files specified with an .include statement. As it reads the files, sreport performs basic syntax error checking, and, if error-free, stores the report specification in the Reports Catalog of the database you specify.

Note: If you update the text file specified in an .include statement, you must execute the sreport command against each report specification that contains the .include statement. This procedure is required in order for the report specification to reflect the changes in the text file.

If the report specification contains syntax errors, sreport prints appropriate error messages. If a report in the text file has the same name as an existing report in the Reports Catalog, the older report definition is replaced. If no prior report exists, Report-Writer adds the report to the Reports Catalog. You can then use the specifications to run a report using either the report command or the Go option from the RBF catalog frame.

If you use sreport to save a report created in RBF, it can strip out many of the RBF formatting statements, which makes it easier to edit the report specification in Report-Writer. You cannot, however, edit the saved report specification in RBF.

The sreport command has the following format:

```
sreport dbname |vnode::dbname[/server_class] filename [-s] [-uusername]
[-Ggroupid]
```

dbname

Specifies the name of the database that is to contain the report, and the *vnode* and *server_class*, if required, as described in Standard Flags and Parameters (see page 13).

filename

Specifies the name of a text file containing report-formatting commands for one or more reports. If you do not specify a full directory path name, Report-By-Forms assumes the current directory.

Report-By-Forms prompts you for this parameter if you do not include it on the command line.

The default extension for this file is .rw.

-s

Suppresses status messages.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

sreport Example

This command stores report definitions in the repdef.rw file into the Reports Catalog of the mydb database:

```
sreport mydb repdef.rw
```

starview Command—Start StarView

The starview command invokes StarView, is a simple forms-based program that helps you manage your distributed databases using Ingres Star. StarView allows access to multiple databases simultaneously. For more information on StarView, see the *Ingres Star User Guide*.

The StarView program operates the same way as forms-based Ingres tools. For a complete explanation of using forms, see the *Character-based Querying and Reporting Tools User Guide*.

The starview command has the following format:

```
starview [vnode::][distdbname][/star]
```

vnode

Specifies a remote vnode name, if you want to run StarView against a distributed database on a remote node.

***distdbname*/star**

Specifies a distributed database name. If specified, the Node Status and LDB Types frame is displayed. If you invoke StarView without specifying a database name, the opening StarView main frame is displayed. The /star is the server class.

starview Example

This command starts StarView to manage distributed databases on the remote node new_york:

```
starview new_york::mystar
```

statdump Command—Print Statistics in iistats and iihistogram Catalogs

The statdump command prints statistics contained in the iistats and iihistograms catalogs of the Standard Catalog Interface.

These views contain statistical information about columns used by the Ingres Query Optimizer as it selects an efficient query processing strategy. The statistical information is typically generated by issuing the optimizedb command.

The statdump command has the following format:

```
statdump [SQL option flags] [-zf filename] [-zc] [-zcpk] [-zdl]
[-zn#] [-zq] [-o filename]
dbname {-r tablename {-a columnname}}|{-xr tablename} [-help]
```

SQL option flags

Passes any of the following SQL option flags. For a complete description of these flags, see sql Command (see page 245).

+U |-U

-u

-c*N*

-t*N*

–ik*N*

-fkxM.N

+w|-w

-x*k*

-zf*filename*

Reads *filename* for all command line arguments. This file must contain one flag only per line. If this flag is specified, no other flags or arguments can appear on the command line—they must appear in the specified file instead.

-zc

Displays statistics on the system catalogs as well as the base tables. If you want statistics for selected system catalogs, use this flag and specify the individual tables with the -r flag. To use this flag, you must be the DBA of the specified database.

-zcpk

Displays statistics from a composite histogram on the primary key structure.

-zdl

Deletes statistics from the system catalogs. When this flag is used, the statistics for the specified tables and columns (if any) are deleted rather than displayed.

-zhex

Produces histogram cell values in hex format, which is useful for seeing how Unicode data is stored.

Warning! A file with histogram data represented in hex format (generated by the -zhex flag) cannot be used as input to the optimizedb -i command. Doing so will result in incorrect histogram data, which will affect the performance of optimization algorithms.

-zn*#*

Displays floating-point values in scientific notation (for example, 9.9999+e9) and sets the precision to the level specified by #. The total width of the displayed number will be equal to the value of the precision level + 7.

-zq

Displays only the information contained in the iistats catalog and not the histogram information contained in iihistograms (quiet mode).

-o *filename*

Directs output to the file specified by *filename*. The resulting file is an ASCII file whose content is identical to the information normally sent to the terminal screen.

The resulting file can be used as input to the optimizedb command (see page 192) on the -i flag.

dbname

Specifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-r*tablename*

Produces statistics for the specified tables only. If omitted, then statistics for all tables are produced.

The table name can be qualified with a valid schema name in the format schema**.**tablename, as described in Schema Qualifier (see page 18).

If the table cannot be found, a warning message is printed and processing continues.

-xr*tablename*

Specifies tablenames to be excluded from processing by statdump. Except for these, statistics for all columns in all tables in the database are produced.

-a*columnname*

Produces statistics for the specified columns only. To specify individual columns you must specify the table name with the -r flag. If column names are omitted, then all columns of the specified table are processed.

If the column cannot be found, a warning message is printed and processing continues.

-help

Displays command syntax online.

Note: The combination of -r*tablename* and -xr*tablename* parameters and of -xr*tablename* and -a*columnname* parameters is not permitted.

statdump Examples

This command prints the statistical information for all columns in the employee table in the empdata database:

```
statdump empdata -remployee
```

This command prints the information in the iistats system table only, for all columns in all tables of the empdata database:

```
statdump -zq empdata
```

This command deletes statistics for all columns in the employee table:

```
statdump -zdl empdata -remployee
```

syscheck Command—Display and Verify System Resources

Permission required: Installation owner, system administrator.

The syscheck command displays process and system resources and verifies that there are enough resources to run Ingres as currently configured.

If resources are sufficient, syscheck prints a confirming message and continues. If resources are insufficient, syscheck displays the resources needed, prints an error message, and exits with an error status.

Ingstart calls syscheck after the system is configured and before starting the servers, so you typically will have no need to call this command directly. However, if system resources have changed since installation, you can use this command to see if you are reaching operating system resource limits that might cause the system to fail.

If you use this command, run it after you have configured your system because syscheck reads the locking and logging parameters before it checks for resources.

Note: The syscheck command returns limited information on Windows and VMS environments.

The syscheck command has the following format:

```
syscheck [-v] [-ofilename] [-help]
```

-v

Displays messages on all (not just the insufficient) resources (verbose mode).

-o*filename*

Indicates that syscheck output is to go to the specified *filename*.

-help

Displays command syntax online.

sysmod Command—Modify System Catalogs to Current Storage Structure

Permission required: DBA or system administrator.

The sysmod command modifies the system catalogs of a database to their currently defined storage structure. Doing so removes overflow and deleted pages, which results in accelerating query processing. You can run sysmod on all or specified system catalog tables.

The sysmod operation requires exclusive access to the database.

The sysmod command has the following format:

```
sysmod dbname [/server_class] {tablename} [-f product {product}]
[-page_size=n][+w|-w]
```

dbname

Specifies the name of the database, and if required, the *server_class*, as described in Standard Flags and Parameters (see page 13). Do not specify the *server_class* as **/star** if the database is an Ingres Star distributed database.

tablename

Specifies individual tables to be modified by sysmod. The tables can be Ingres Star standard catalogs or Ingres Star-specific system catalogs. If omitted, all tables in the database are processed.

The table name can be qualified with a valid schema name in the format schema**.**tablename, as described in Schema Qualifier (see page 18).

-f *product*

Specifies the user interface products for which you want to modify system tables. Allowable product names are ingres, ingres/dbd, vision, and windows_4gl, as described in Standard Flags and Parameters (see page 13). If you omit this parameter, all user interfaces are processed.

Note: You cannot specify individual user interface catalogs; when you specify the product parameter, all catalogs are processed for that user interface product.

-page_size=*n*

Permits modifying the existing system catalogs with a different page size: Specify page_size=n, where n is one of 2048, 4096, 8192, 16384, 32768, 65536.

Example: SYSMOD test –page_size=4096

+w|-w

Directs sysmod to wait (+w) or not wait (-w) until the database is free before executing. The default is -w.

On VMS, this flag is not valid in batch mode.

tables Command—Start the Tables Program

The tables command starts the Tables program, a forms-based interface for creating, destroying and examining tables.

For a complete description of the Tables program, see the *Character-based Querying and Reporting Tools User Guide*.

The tables command has the following format:

```
tables dbname|vnode::dbname[/server_class][-e] [-uusername] [-Ggroupid]
```

dbname

Specifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-e

Invokes the tables program in empty mode. The catalog is initially displayed empty, so that the user can enter specific names of database objects.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

tables Example

This command invokes the Tables utility with an empty initial catalog of the emp database for the effective user emma:

```
tables emp -e -uemma
```

unextenddb Command—Unextend a Database Location

The unextenddb command unextends a database location that was previously extended by the extenddb command, and deletes the entry for the location in the configuration file. The location can then be used again by the extenddb command. If tables or files exist in the location, an error message is issued and the database cannot be unextended at that location.

Note: After unextending a database location, you should checkpoint the database. Previous checkpoints cannot be used because they reference a location that is no longer accessible to the database.

The unextenddb command has the following format:

```
unextenddb -llocation [dbname...] [-Udata,work|awork] [-P] [-uuser]
```

-l*location*

Specifies the name of the location to be unextended.

***dbname*...**

Specifies the list of databases to be operated on by unextenddb, and if required, the *vnode*, as described in Standard Flags and Parameters (see page 13).

-Udata,work|awork

Specifies the use of the extended location. Valid uses include database, work, and auxiliary work.

-P

Indicates the password if the session requires one.

-u*user*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

unloaddb Command—Create Command Files for Unloading and Reloading a Database

The unloaddb command creates command files that the DBA uses to unload the data from a database and reload the data into a new, empty database.

Use unloaddb when a database must be totally rebuilt, or for checkpointing the database. The unloaddb command unloads all objects in the database, including tables, views, integrity constraints, permissions, forms, graphs, and report definitions.

Two command files are created:

- Unload file—contains commands to read sequentially through the database, copying every user table into its own file in the named directory.
- Reload file—contains commands to load a new, empty database with the information contained in the files created by the unload file.

On Windows the file names are unload.bat and reload.bat. On UNIX, the file names are unload.ing and reload.ing.

The DBA must execute these files to accomplish the unloading and reloading of the database. It is important that the database be recreated with the reload file before doing any work (for example, creating tables, forms, and reports) in the new database.

The unloaddb command uses a version of the copydb command to generate the copy commands in the unload and reload files. Consequently, all limitations of the copydb command apply to the unloaddb command.

Note: If overflow occurs, you may need to edit the unload and reload files to specify another flag, for example, N instead of F in the default floating point specification.

Note: To optimize performance, run the sysmod and optimizedb commands after recreating the database.

Note: When unloaddb is run from an Ingres 2006 Release 2 or later installation against an older version of Ingres, the command file generated will contain the data type INGRESDATE or ANSIDATE instead of DATE for any date columns in create table statements.

For additional information on unloading a database, see the *Database Administrator Guide*.

The unloaddb command has the following format:

```
unloaddb dbname|vnode::dbname[/server_class] [-c] [-ddirname] [-source=dirname]
[-dest=dirname] [-P] [-uusername] [-Ggroupid] [-group_tab_idx]
[-parallel] [-journal] [-with_sequences] [-no_rep]
```

dbname

Specifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-c

Creates printable data files, which is useful for transporting databases between computer systems whose internal representations of non-ASCII data differ.

Unloaddb cannot create printable files if (1) binary data is stored in varchar columns, or (2) tables contain user-maintained logical keys.

For nchar, nvarchar, or long nvarchar columns or when the installation character set is UTF8, the data files generated are in UTF-8 encoding.

The UTF-8 encoded data files containing data from char, varchar, or long varchar columns can be reloaded only into installations installed with the UTF8 character set.

-d*dirname*

Stores the unload and reload files in the location specified by *dirname* instead of the default current directory. The specification can be either a full or relative directory specification.

The *dirname* must not be the actual database directory, because the files created by unloaddb may have the same names as the tables in the database. The actual database directory is: $II_DATABASE/ingres/data/default/dbname. (On VMS, the directory is: II_DATABASE:[INGRES.DATA.DBNAME].)

-source=*dirname*

Specifies the source directory from which the database will be reloaded. An empty *dirname* specification ("") denotes the current directory. The -source specification overrides a -d specification for the reload file.

If a source is specified without a destination (no -d or -dest), then the default unload directory is used.

The source directory specification is not checked for validity or existence. This allows the scripts to be moved to another machine for reloading.

-dest=*dirname*

Specifies the destination directory into which the database will be unloaded. An empty *dirname* specification (".") denotes the current directory. The -dest specification overrides a -d specification for the unload file.

If a destination is specified without a source (no -source) then the default reload directory is used.

The destination directory specification is not checked for validity or existence. This allows the scripts to be moved to another machine for unloading.

-P

Prompts for password if the session requires a password.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("**-**G*groupid*").

-parallel

Creates indexes using the parallel index creation syntax (to build multiple indexes concurrently).

-journal

Replaces the "set nojournaling" statement in the unload scripts with the "set journaling" statement, and disables specifying the "with nojournaling" option on each create table statement in the unload script.

-with_sequences

Print statements related to sequences only.

-group_tab_idx

Builds indexes in the command file immediately after the respective table creation. Without this flag, all indexes are created for all tables toward the end of the command file. The usermod command uses this flag to limit the loss of non-persistent indexes if it encounters a failure.

-no_rep

Does not write Ingres Replicator objects (tables, indexes, events, procedures) of a replicated database to the unload file.

unloaddb Example: Unload and Reload a Database

The following commands unload and reload the empdata database:

Windows:

```
cd\mydir\backup
unloaddb empdata
unload
destroydb empdata
createdb empdata
reload
sysmod empdata
```

UNIX:

```
cd /mydir/backup
unloaddb empdata
unload.ing
destroydb empdata
createdb empdata
reload.ing
sysmod empdata
```

VMS:

```
set default [mydir.backup]
unloaddb empdata
@unload.ing
destroydb empdata
createdb empdata
@reload.ing
sysmod empdata
```

unloaddb Example: Unload a Database, Specifying Source and Destination Directories

This command unloads the empdata database, specifying separate source and destination directories:

```
unloaddb empdata -source="misc/loaddir/" -dest="misc/dumpdir"
```

Copy statements in the reload script would have the form:

```
copy emps () from 'misc/loaddir/emps.bob'
```

Copy statements in the unload script would have the form:

```
copy emps () into 'misc/dumpdir/emps.bob'
```

unloaddb Example: Unload a Database from the $HOME Directory

This command unloads the empdata database from the $HOME directory, specifying the current directory as the source and destination directories:

```
unloaddb empdata -source="" -dest=""
```

Copy statements in the reload script would have the form:

```
copy emps () from 'emps.bob'
```

Copy statements in the unload script would have the form:

```
copy emps () into 'emps.bob'
```

upgradedb Command—Upgrade a Database

The upgradedb command installs and upgrades one or all databases in the Ingres installation.

Upgradedb triggers upgradefe. For more information, see the upgradefe command description.

If upgradedb cannot upgrade the user interface catalogs for a database, it prints a warning and marks the database operative. You can then either run upgradefe directly on the database, or rerun upgradedb, specifying the database individually with the *dbname* parameter.

Note: Upgradedb takes a lock on iidbdb (unless you use the -c flag). Make sure rmcmd is shut down (that is, set to zero in CBF) before running upgradedb.

The upgradedb command has the following format:

```
upgradedb dbname |vnode::dbname[/server_class]|-all [-f product {product}] [-c]
[-help]
```

dbname

Specifies the name of the database (one name only) to be upgraded, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-all

Causes upgradedb to operate on all databases in the installation that have not already been upgraded to the new release level. With this flag, upgradedb skips any databases already at the current release level.

Note: You can specify either *dbname* or all, but not both.

-f *product*

Specifies the user interface products for which you want to upgrade the database. Allowable product names are ingres, ingres/dbd, vision, windows_4gl and nofeclients, as described in Standard Flags and Parameters (see page 13). If you omit this parameter, all Ingres tools for the database are processed.

Note: We advise, in most cases, to use the same -f command to upgrade the database that was used to create it initially. If you are upgrading a pre-6.3 database, the -fingres command upgrades your existing user interface catalogs without creating the catalogs for the Ingres tool products that were new with 6.3 (Vision and Windows 4GL).

-c

Runs upgradedb concurrently from multiple sessions. To use this feature, run upgradedb on iidbdb before concurrently upgrading the user database. Specify flag -c in every upgradedb session.

-help

Displays command syntax online.

upgradefe Command—Install and Upgrade Tool Catalog Definitions

The upgradefe command installs and upgrades the catalogs required by Ingres tools. You must execute the upgradefe command after installing a new version of any Ingres tool, if the new version requires changes to the catalog definitions.

The upgradefe command has the following format:

```
upgradefe dbname |vnode::dbname[/server_class] {product} [-b] [-vversion] [-s]
[-c] [-uusername]
```

dbname

Specifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

product

Specifies the Ingres tool products for which you want to upgrade the database. Allowable product names are ingres, ingres/dbd, vision, and windows_4gl, as described in Standard Flags and Parameters (see page 13). If you omit this parameter, all user interfaces for the database are processed.

-b

Installs the modules required to support the product Ingres. (Specifying the -b flag is equivalent to specifying the product ingres.)

-v*version*

Specifies the version number of the product to be installed; if the -v flag is not specified, the highest known version is installed.

-s

Suppresses messages from upgradefe.

-c

Allows multiple concurrent connections to iidbdb without taking exclusive locks.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13). If you want to upgrade a database you do not own, you must use the -u flag to specify the user name of the DBA.

upgradefe Examples

This command installs catalogs for OpenROAD:

```
upgradefe mydb windows_4gl
```

This command installs catalogs for the base tools and Vision:

```
upgradefe mydb ingres vision
```

usermod Command—Modify Tables to Currently Defined Storage Structure

The usermod command modifies the user-defined tables of a database to their currently defined storage structure and recreates any secondary indexes that are currently defined.

Similar to the sysmod command for system catalogs, usermod is a useful utility for maintaining user tables. Running the usermod utility on a regular basis or when the table has excess overflow pages improves the performance of query processing.

You can run usermod on the whole database or on specified tables. If no specific tables are specified, all the tables belonging to the user are modified. The usermod command writes its files to the II_TEMPORARY directory.

Unless the -online flag is specified, usermod takes exclusive locks on each table in turn for the time it takes to modify the table, and also takes an exclusive lock during the rebuilding of the various secondary indexes.

Caution! Should the process fail for any reason, some non-persistent secondary indexes may not be rebuilt. A message will be displayed indicating the location of the SQL script used to perform the modify, which can be examined to determine any missing indexes.

The usermod command has the following format:

```
usermod dbname /vnode::dbname[/server_class] [-uusername] [tables] [-online]
[-noint] [-repmod [+w|-w]]
```

dbname

Specifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13). Do not specify the *server_class* as /star if the database is an Ingres Star distributed database.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

tables

Specifies individual tables to be modified by usermod. The table names should be separated by spaces. They can be Ingres Star standard catalogs or Ingres Star-specific system catalogs. If omitted, all tables in the database are processed. The table name can be qualified with a valid schema name in the format *schema.tablename*, as described in Schema Qualifier (see page 18).

-online

Performs the modification online. The –online option is equivalent to the "with concurrent_updates" option on the modify statement in SQL.

Note: During an online modify operation, normal read and update access to the table is permitted, except for a brief period at the end, where exclusive access to the table is required.

-noint

Runs the usermod command uninterrupted, even if there are errors.

-repmod [+w|-w]

Runs the repmod utility (see page 229) to modify the replicator catalogs also. By default, the usermod command does not modify the replicator catalogs. To use the –repmod option, the database must be replicated.

The repmod operation takes exclusive lock on the database. Use the +w or –w flags on –repmod to specify whether repmod will wait for the database to be free if it is in use. The default is -w (do not wait).

vcbf Command—Start Configuration Manager

Permission required: Access to the directory where the utility is located.

The vcbf command starts the Configuration Manager, a graphical user interface for configuring the Ingres installation.

The Configuration Manager displays current values of the server parameters and provides menu for changing them. With the Configuration Manager you can configure various components of the installation, select which databases a DBMS Server can access, reformat transaction log files and enable or disable dual logging, reconfigure protocol accesses for the Communications Server, set a new value of any configuration parameter or restore the default, automatically calculate configuration parameters derived from other parameters, protect any derived parameter from further change, run a system check for sufficient resources on a new configuration, and view a log of all configuration changes.

The vcbf command has the following format:

```
vcbf
```

vcda Command—Start the Visual Configuration Differences Analyzer

Permission required: Installation owner, DBA, privileged user.

The vcda command starts the Visual Configuration Differences Analyzer (VCDA) utility, which is a graphical user interface used by the DBA to compare configuration snapshots of Ingres installations.

The vcda command has the following format:

```
vcda [-c|snapshotfile1] [snapshotfile2]
```

-c

Indicates the current, local Ingres installation.

snapshotfile1

Specifies a file containing a configuration snapshot of an Ingres installation.

snapshotfile2

Specifies a second file containing another configuration snapshot of an Ingres installation.

vdba Command—Start Visual DBA

Permission required: Installation owner, DBA, privileged user.

The vdba command starts the Visual DBA utility, which is a graphical user interface used by the DBA to administer an Ingres installation. The vdba utility can be used to manage a local or remote Ingres installation.

The vdba command has the following formats:

```
vdba environmentname.cfg
```

or

```
vdba /c [maxapp] [maxwin] [nonodeswindow] [windowdesc {,windowdesc}...]
```

***environmentname*.cfg**

Specifies a configuration file that defines a Visual DBA environment previously saved with Visual DBA.

Note: If you specify this file, no other parameters are allowed on the command line.

/c

Provides a means of invoking Visual DBA with a list of parameters.

maxapp

Maximizes the Visual DBA application.

maxwin

Maximizes the MDI windows in Visual DBA.

nonodeswindow

Does not display the Virtual Nodes toolbar and window.

windowdesc

Specifies the type of window to open on startup. Valid values are dom, sql, monitor, and dbevent.

The syntax for each *windowdes* is as follows:

```
dom|sql|monitor|dbevent [nodename] [/server_class] [-uusername] [objecttype objectidentifier]
```

where:

dom|sql|monitor|dbevent

Specifies the type of window to be opened.

nodename

Specifies the node where the window is to open. No node designates the local node.

/server_class

Specifies the server class (allows work on Enterprise Access products)

-u*username*

Specifies the user to impersonate.

vnode

Specifies the remote node for which specified windows should be opened, as described in Standard Flags and Parameters (see page 13).

objecttype objectidentifier

Places the selection on the corresponding object: In dom and monitor windows, expands appropriate branches and places the selection of the corresponding object.

In sql and dbevent windows, only a database can be specified (it becomes the active database).

Valid values for *objecttype* are: database, table, view, procedure, user, group, role, location, or server.

Valid values for *objectidentifier* are:

serverno (for servers) (in monitor windows)

objectname (if not a child branch from a database)

dbname/*objectname* (if child from a database). Schema prefixes are acceptable.

Note: The last *windowdesc* becomes the topmost window in Visual DBA.

vdba Examples

This command invokes Visual DBA without the Virtual Nodes window, maximized on the screen, with one dom window opened for the local node:

```
vdba /c nonodeswindow maxwin dom
```

This command invokes Visual DBA, maximized on the screen, with both the performance monitor window and a dom window opened for the local node:

```
vdba /c maxapp monitor, dom
```

This command invokes a saved Visual DBA environment:

```
vdba gateway.cfg
```

vdbamon Command—Start Visual Performance Monitor

Permission required: Installation owner, DBA, privileged user.

The vdbamon command invokes the Visual Performance Monitor, a function of Visual DBA.

The vdbamon command has the following format:

```
vdbamon [-node=<node name> [/serverclass]] [-uusername] [-maxapp]]
```

-node=*nodename*

Specifies the node where the window is to open. If no node is specified, the local node is used.

server_class

Specifies the server class (allows work on Enterprise Access products)

-u*username*

Specifies the user to impersonate

-maxapp

Maximizes the window

vdbasql Command—Start Visual SQL

Permission required: Installation owner, DBA, privileged user.

The vdbasql command invokes Visual SQL, a function of Visual DBA.

The vdbasql command has the following format:

```
vdbasql [-node=<node name> [/serverclass] [-database=dbname]] [-uusername]
[-maxapp]
```

-node=*nodename*

Specifies the node where the window is to open. If no node is specified, the local node is used.

server_class

Specifies the server class (allows work on Enterprise Access products)

-database=*dbname*

Specifies the database to be accessed

-u*username*

Specifies the user to impersonate

-maxapp

Maximizes the window

vdda Command—Start the Visual Database Objects Analyzer

Permission required: Installation owner, DBA, privileged user.

The vdda command starts the Visual Database Objects Differences Analyzer (VDDA) utility, which is a graphical user interface the DBA can use to compare the definitions of Ingres database objects.

The vdda command has the following format:

```
vdda
```

verifydb Command—Clean Up Databases

The verifydb command performs clean up operations on one or more databases in an installation.

The command can do the following:

- Delete unneeded disk files in a database directory
- Delete temporary and expired tables
- Remove all references to a specified table from the DBMS system catalogs

This command requires exclusive access to databases. Verify that there are no active sessions in the DBMS before continuing. If users are connected to the database, a runtime error is displayed. Shut down processes that maintain database connections, rmcmd and icesvr, before using this command.

Verifydb logs all of its actions to the terminal screen. It also logs to a verify log file, unless the -n (nologging) flag is used. The default log file is iivdb.log and is used unless another name is specified with the -lf option.

Note: Verifydb always outputs the log file to the II_CONFIG location. If II_CONFIG is not defined, it outputs to location: II_SYSTEM/ingres/files.

If the log file does not exist when you execute verifydb, the system creates it. If it does exist, verifydb appends to it. Since this file grows each time you execute verifydb with this log file, you should delete it occasionally to save disk space.

The verifydb command has the following format:

```
verifydb -mmode -sscope -ooperation [-n | -lflogfilename] [-v] [-uusername]
```

-m*mode*

Specifies the mode in which verifydb executes. The *mode* can be one of the following:

report

Directs verifydb to log its findings. Use report mode if you want verifydb to only log, rather than actually delete, the tables or files that it finds.

run

Directs verifydb to perform the specified operation and log all actions that it performs.

runinteractive

Directs verifydb to prompt the user for confirmation before each action is taken. If the user responds negatively to a prompt, verifydb skips that action and goes on to the next.

runsilent

Tells verifydb to perform the specified operations but turns off the logging to the terminal. (Logging to the log file continues.)

-s*scope*

Tells verifydb to perform the operation only on the specified databases. All databases must have the same owner. You can specify up to 10 databases.

Scope can be any of the following:

dbname "*dbname* **|***vnode***::***dbname***[/***server_class***]{***dbname* **|***vnode***::***dbname***[/***server_class***]}"**

dba

Directs verifydb to operate on all databases for which the user is the DBA or for all databases owned by the DBA specified by the -u flag.

installation

Directs verifydb to operate on all operative databases. You must be a privileged user to use this qualifier.

-o*operation*

Specifies the operation to be performed. If report mode is specified, the files or tables found are not actually deleted, but only logged. The options for operation are:

accesscheck

Checks each database specified by the scope and returns a message that says whether the server can connect to the database and, if not, the reason. When using this option, you must also specify report mode (-mreport).

You must be either a DBA or a privileged user to use this option. If you are a DBA and specify a scope of dbname, you must be the DBA of all the listed databases. If you use a scope of dba, verifydb checks all the databases for which you are the DBA. If you use a scope of installation, you must be a privileged user, and accesscheck checks all databases in the installation.

Additionally, if you are a privileged user, you can use the -u flag to run this option as another user.

purge

Directs verifydb to delete all disk files in the database directory that are no longer required. This operation is a combination of temp_purge and expired_purge.

temp_purge

Tells verifydb to search for and delete all temporary tables from the database.

expired_purge

Directs verifydb to search for and delete all expired tables from the database.

drop_table "*tablename*"

Tells verifydb to remove all references to a specified table from the DBMS system catalogs. If you specify this option, you must use the dbname option for the -s flag.

table "*tablename*"

Checks the specified tables and reports any inconsistencies found, making recommendations to repair those inconsistencies. The table operation cannot be used on core system catalogs. Secondary indexes can be checked but cannot be repaired. A table lock is taken during verifydb table operations, but a database lock is not taken. Use this option only when you are using report mode (-mreport).

This operation also verifies referential integrity between the internal pointers for long data types stored in base table records and the extension table records they point to. Any inconsistencies are reported.

Caution! *Using this option when you are in any run mode is not supported unless you are receiving assistance from customer support and are advised to do so; it can have severe, unexpected results.*

xtable "*tablename*"

Functions like the table option, however xtable uses a stricter patch algorithm, which guarantees data integrity—with the risk that some valid data may be discarded. Use this option only when you are using report mode (-mreport).

Caution! *Using this option when you are in any run mode is not supported unless you are receiving assistance from customer support and are advised to do so; it can have severe, unexpected results.*

dbms_catalogs

Checks the dbms catalogs and reports any inconsistencies found, making recommendations to repair those inconsistencies. Use this option only when you are using report mode (-mreport).

Caution! *Using this option when you are in runinteractive mode is not supported unless you are receiving assistance from customer support and are advised to do so; it can have severe, unexpected results. This operation is not supported in run modes other that runiteractive.*

force_consistent

Permits entry into a database that is inconsistent. This does not fix the problem with the database; it merely allows you to force the database to act as if it were in a consistent state. This can be very dangerous if used against a production database. Hidden data damage may render one or more tables in the database unrecoverable at some time in the future. Use this option only when you are using report mode (-mreport).

Caution! *Using this option when you are in any run mode is not supported unless you are receiving assistance from customer support and are advised to do so; it can have severe, unexpected results.*

refresh_ldbs

Directs verifydb to assure that a distributed database correctly reflects the release level of all remote databases that contain objects registered to the distributed database. It is recommended that you run this operation on a distributed database after you run upgradedb on any of the remote databases accessed by the distributed database. For additional information on using this parameter, see the *Ingres Star User Guide*.

The distributed databases are specified by the **-s***scope* parameter. For the refresh_ldbs option only, verifydb skips all non-distributed databases and processes only distributed databases. (In all other cases, verifydb processes only non-distributed databases.)

-n

Turns off logging to the log file (nolog mode). Logging to the terminal continues.

-lf*logfilename*

Specifies an alternate log file (in the II_CONFIG location) to which verifydb is to log activity. When using this flag, the -n flag is not permitted.

-v

Provides additional dialog messages when performing the verifydb operation (verbose mode). This flag applies only for table operations.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

verifydb Examples

This command cleans up all databases for which you are the DBA, removing all unrequired disk files, and logging all verifydb operations performed:

```
verifydb -mrun -sdba -opurge
```

This command runs in report mode, looking for expired tables in the database teach_examp:

```
verifydb -mreport -sdbname "teach_examp" -oexpired_purge
```

This command runs verifydb as the user fredk against all the databases for which fredk is the DBA, deleting temporary and expired tables:

```
verifydb -mrun -sdba -opurge -ufredk
```

This command drops references to the table new_benefits in the database new_employee:

```
verifydb -mrun -sdbname "new_employee" -odrop_table "new_benefits"
```

This command runs consistency checks on the DBMS catalogs for the iidbdb database. The command is run in report mode.

```
verifydb -mreport -sdbname "iidbdb" -odbms_catalogs
```

This command runs consistency checks on the DBMS catalogs for all databases that you own, with output going to the alternate log file checkdbs.log. The command is run in report mode.

```
verifydb -mreport -sdba -odbms_catalogs -lfcheckdbs.log
```

vifred Command—Start the Visual Forms Editor

The vifred command invokes the Visual Forms Editor (VIFRED), a forms-based interface for editing the appearance of a form. For a complete description of this system, see the Character-*based Querying and Reporting Tools User Guide*.

The vifred command has the following format:

```
vifred dbname | vnode::dbname[/server_class] [[-f]form | -ttablename |
-jjoindef][-e] [-uusername][-Ggroupid]
```

dbname

Specifies the name of a database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

[-f]*form*

Specifies an existing form created with VIFRED. The -f flag itself is optional, because VIFRED assumes it is a form unless the -t or -j flag is used.

-t*tablename*

Specifies a table name on which VIFRED is to create a default form.

The table name can be qualified with a valid schema name in the format *schema*.*tablename*, as described in Schema Qualifier (see page 18).

-j*joindef*

Specifies a JoinDef on which VIFRED is to create a default form.

-e

Starts VIFRED with an empty table field in the Catalog frame. The user can then access the desired form directly by entering its name in the table field, or use pattern matching to retrieve a range of names for selection.

-u*username*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

vifred Examples

This command starts VIFRED in its initial Forms Catalog frame in the employee database (this is equivalent to invoking Ingres Menu with the employee database and selecting Forms):

```
vifred employee
```

This command starts VIFRED to edit the finance form in the employee database (this is equivalent to selecting Forms from Ingres Menu, then Edit with the cursor on the finance form):

```
vifred employee finance
```

This command starts VIFRED with a default form for the lastname table (this is equivalent to selecting Forms from Ingres Menu, then Create and Table):

```
vifred employee -t lastname
```

vision Command—Start Vision

The vision command invokes the Vision application generator program.

The vision command has the following format:

```
vision dbname |vnode::dbname[/server_class] [applname] [-w] [+wopen] [-uusername]
[-Ggroupid]
```

dbname

Specifies the name of the database, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

applname

Specifies the name of a Vision application to be viewed or edited.

-w

Checks for conflicts between an application's procedure names and QUEL function names. If you define a procedure that has the same name as a QUEL function—for example, date()—your date procedure supersedes the QUEL function.

+wopen

Issues a warning if the preprocessor detects an embedded SQL statement that does not follow OpenSQL syntax. (For more information about OpenSQL, see the *OpenSQL Reference Guide*.) Warnings do not halt or affect the success of compilation.

Note: This flag does not validate the statement syntax for any SQL gateway whose syntax is more restrictive than that of OpenSQL.

-u***username***

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

On VMS, enclose this parameter in double quotation marks ("-G*groupid*").

vmsinstal Command—Install Ingres on OpenVMS

Valid on OpenVMS.

Permission required: Installation owner.

The vmsinstal utility performs the initial installation of your Ingres release in the OpenVMS environment. Use this utility when installing Ingres for the first time and when updating to a new version.

If you are updating an existing installation, shut it down using ingstop before you run the install program.

The vmsinstal command has the following format:

```
@vmsinstal distribution_medium
```

xmlimport Command—Import XML Data into Ingres

The xmlimport utility imports into the Ingres database any XML data file containing Ingres table data that conforms to Ingres dtd.

This utility is useful to import an XML file generated by genxml (see page 92) into an Ingres database.

You can have several table definitions and index definitions in the XML file. When xmlimport runs, the tables and indexes specified in the XML file are created in the provided database, and the data is uploaded.

Internally, xmlimport parses the XML file to generate an SQL script from the metadata information in the XML file and for the data files for each table's data. The xmlimport utility then runs the SQL script to create tables and upload the data from the data files. Files are created in the temp directory and deleted when the script has been run, except when the -debug flag is specified.

The xmlimport command has the following format:

```
xmlimport dbname [uuser] [P] [-GgroupID] [-debug] xmlfile
```

dbname

Specifies the name of the database being exported, and if required, the *vnode* and *server_class*, as described in Standard Flags and Parameters (see page 13).

-u*user*

Specifies the effective user for the session, as described in Standard Flags and Parameters (see page 13).

-P

Specifies the password for the session, if required.

-G*groupid*

Specifies a group identifier, as described in Standard Flags and Parameters (see page 13).

-debug

Leaves the generated XML file and the data files in the temp location. By default, the files in this location are deleted.

xmlfile

Specifies the name of the XML file that needs to be imported into the database.

The xmlimport utility validates the XML file against the generic Ingres dtd. If the ingres dtd is External, it should by default be at the same location as the XML file. If Ingres is referred to the ingres.dtd in the $II_SYSTEM/ingres/files area, the dtd should be present in the $II_SYSTEM/ingres/files directory.

If the tables and indexes are already in the database, an error displays and the data is appended to the existing table.

xmlimport Example

This command imports an XML file named xmlout.xml into an Ingres database testdb:

```
xmlimport testdb xmlout.xml.
```

The xmlimport utility parses xmlout.xml and then creates the tables and indexes defined in the xmlout.xml file in testdb database.

Index

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

X