DB2 IBM Informix

Version 10.0

IBM Informix Dynamic Server Administrator's Reference

G251-2268-01

DB2® IBM Informix

IBM

Version 10.0

IBM Informix Dynamic Server Administrator's Reference

G251-2268-01

Note:
Before using this information and the product it supports, read the information in “Notices” on page G-1.

Second Edition (December 2005)

Contents

Introduction

In This Introduction

This introduction provides an overview of the information in this manual and describes the conventions it uses.

About This Manual

This manual provides reference material for IBM Informix Dynamic Server. It contains comprehensive descriptions of configuration parameters, the system-monitoring interface (SMI) tables in the **sysmaster** database, the syntax of database server utilities such as **onmode** and **onstat**, logical-log records, disk structures, event alarms, and unnumbered error messages. This manual has two companion volumes, the *IBM Informix Administrator's Guide* and the *IBM Informix Performance Guide*.

This section discusses the organization of the manual, the intended audience, and the associated software products that you must have to use the administrative utilities.

Types of Users

This manual is written for the following users:

- Database administrators
- System administrators
- Performance engineers

This manual is written with the assumption that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with database server administration, operating-system administration, or network administration

If you have limited experience with relational databases, SQL, or your operating system, refer to the *IBM Informix Getting Started Guide* for your database server for a list of supplementary titles.

Software Dependencies

This manual is written with the assumption that you are using IBM Informix Dynamic Server or IBM Informix Dynamic Server with J/Foundation, Version 10.0, as your database server.

Assumptions About Your Locale

IBM Informix products can support many languages, cultures, and code sets. All the information related to character set, collation, and representation of numeric data, currency, date, and time is brought together in a single environment, called a Global Language Support (GLS) locale.

The examples in this manual are written with the assumption that you are using the default locale, **en_us.8859-1**. This locale supports U.S. English format conventions for date, time, and currency. In addition, this locale supports the ISO 8859-1 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *IBM Informix GLS User's Guide*.

Demonstration Database

The DB–Access utility, which is provided with your Informix database server products, includes one or more of the following demonstration databases:

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in IBM Informix manuals are based on the **stores_demo** database.
- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines.

For information about how to create and populate the demonstration databases, see the *IBM Informix DB–Access User's Guide*. For descriptions of the databases and their contents, see the *IBM Informix Guide to SQL: Reference*.

The scripts that you use to install the demonstration databases reside in the **$INFORMIXDIR/bin** directory on UNIX and in the **%INFORMIXDIR%\bin** directory on Windows.

New Features in Dynamic Server, Version 10.0

The following table provides information about the new features for IBM Informix Dynamic Server, Version 10.0, that this manual covers. For a description of all new features, see the *IBM Informix Getting Started Guide*.

New Features	Reference
Support for Distributed Relational Database Architecture (DRDA) protocol	You can connect to DB2 clients through DRDA. Some onstat commands include the DRDA thread name and an indicator to distinguish between SQLI and DRDA sessions.
ALRM_ALL_EVENTS	Use this new event-alarm parameter if you want the event-alarm program to operate for all events that are logged in the MSGPATH, instead of only noteworthy events. See "ALRM_ALL_EVENTS" on page 1-13 and "Setting ALRM_ALL_EVENTS" on page C-1.
Buffer pool size in **onstat** utility output	The buffer pool size in the output from **onstat** commands is now indicated in bytes (the page size). See Chapter 14, "The onstat Utility," on page 14-1.
BUFFERPOOL	The BUFFERPOOL configuration parameter allows you to specify different page size buffer pools. Information that was specified with the BUFFERS, LRUS, LRU_MAX_DIRTY, and LRU_MIN_DIRTY configuration parameters prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter. See "BUFFERPOOL" on page 1-14.
4 4 4 4 DB_LIBRARY_PATH	Use this configuration parameter to limit the paths from which UDRs can be run to those paths that you specify. See "DB_LIBRARY_PATH" on page 1-21.

	New Features	Reference
3	DBCREATE_PERMISSION	DBCREATE_PERMISSION restricts the permission to create databases to the specified user. You can include multiple copies of DBCREATE_PERMISSION in the ONCONFIG file to give additional users permission to create databases. For more information see "DBCREATE_PERMISSION" on page 1-21.
	DISABLE_B162428_XA_FIX	The DISABLE_B162428_XA_FIX configuration parameter ensures compatibility with applications that require the default behavior of Dynamic Server version 9.4 or earlier. This configuration parameter immediately frees all global transactions after a transaction rollback.
	DRAUTO	The DRAUTO configuration parameter determines how a secondary database server reacts to an HDR failure. See "DRAUTO" on page 1-29 and the *IBM Informix Dynamic Server Administrator's Guide*.
	DRIDXAUTO **onmode -d index** **onmode- d idxauto**	Used to specify whether the primary High-Availability Data Replication (HDR) server automatically starts index replication if the secondary HDR server detects a corrupted index. The **-d idxauto** option dynamically changes DRIDXAUTO. The **-d index** option forces index replication. See "DRIDXAUTO" on page 1-28 and "Replicate an Index with Data-Replication" on page 10-10.
	DS_NONPDQ_QUERY_MEM	Use this configuration parameter to increase the amount of sort memory that is a available for a query that is not a Parallel Database Query (PDQ). See "DS_NONPDQ_QUERY_MEM" on page 1-33, the *IBM Informix Dynamic Server Administrator's Guide*, and the *IBM Informix Dynamic Server Performance Guide*.
3	DUMPDIR	The default value of the DUMPDIR configuration parameter in the onconfig.std file is `/usr/informix/tmp` on UNIX and `%INFORMIXDIR%\tmp` on Windows. If the DUMPDIR configuration parameter is not in the ONCONFIG file, the `$INFORMIXDIR/tmp` directory is used
	FAST_RESTART_CKPT_FUZZYLOG FAST_RESTART_PHYSLOG	These two configuration parameters enable the database server to perform physical logging on fuzzy checkpoints during the roll-forward phase of recovery, thus reducing the time required for recovery. See "FAST_RESTART_CKPT_FUZZYLOG" on page 1-40 and "FAST_RESTART_PHYSLOG" on page 1-40.
4	IFX_EXTEND_ROLE	The default, 1, enables the EXTEND role so that administrators can grant privileges to a user to create or drop a DataBlade UDR that has the EXTERNAL clause. See "IFX_EXTEND_ROLE" on page 1-42 and the *IBM Informix Administrator's Guide*.

New Features	Reference
LISTEN_TIMEOUT MAX_INCOMPLETE_ CONNECTIONS	New configuration parameters for Windows systems that you can use to reduce the incomplete connection timeout period and restrict the number of incomplete requests for connections, thus reducing the risk of a hostile, denial-of-service (DOS) flood attack. See "LISTEN_TIMEOUT" on page 1-43 and "MAX_INCOMPLETE_CONNECTIONS" on page 1-50 and the Security chapter in the *IBM Informix Dynamic Server Administrator's Guide*.
onmode -j	This option is used to change the database server to single-user mode. See "Change Database Server Mode" on page 10-4.
onparams -b	The **-b** flag for **onparams** utility allows you to create a new buffer pool while the database server is running. See "Add a New Buffer Pool" on page 12-4.
Rename a Dbspace, Blobspace, Sbspace, or Extspace with the **onspaces** utility	See "Rename a Dbspace, Blobspace, Sbspace, or Extspace" on page 13-17.
SHMADD SHMVIRTSIZE	These configuration parameters allow you to increase the size of segments for shared memory. See "SHMADD" on page 1-71 and "SHMVIRTSIZE" on page 1-73.
TBLTBLFIRST TBLTBLNEXT	Specifies the first and the next extent size of tblspace **tblspace** in the root dbspace. See "TBLTBLFIRST" on page 1-83, "TBLTBLNEXT" on page 1-84, and "Structure of the Tblspace Tblspace" on page 3-5.
UNSECURE_ONSTAT	The **onstat** commands that show SQL statement text being executed by a session are normally restricted to DBA users. You can set the UNSECURE_ONSTAT configuration parameter to 1 to remove this restriction.

Features From Previous Versions

The following sections provide information about features that were added to previous versions of Informix Dynamic Server.

Features From Dynamic Server, Version 9.4

The following tables provide information about features that were added to IBM Informix Dynamic Server, Version 9.4.

Performance Enhancements

Version 9.4 included several features to help you monitor and improve the performance of your database.

New Features	Reference
CDR_MAX_DYNAMIC_LOGS	For details, refer to your *IBM Informix Dynamic Server Enterprise Replication Guide*.

New Features	Reference
onmode -Y	This new **onmode** option allows you to dynamically change the SQEXPLAIN setting. For more information, see "Dynamically Setting of SET EXPLAIN" on page 10-19.
oncheck -pP	This **oncheck** option has been enhanced to allow users to optionally specify the number of pages to be dumped. For more information, see "Display the Contents of a Logical Page with -pp and -pP" on page 6-14.
> 2 GB chunks, offsets with **onmode -BC 1** and **onmode -BC 2**	This option allows a significantly large increase in the size of chunks, offsets, and total number of chunks. For more information, see page "Allow Large Chunk Mode" on page 10-3.
B-tree scanner with **onstat -C**	This option allows the new B-tree scanner to automatically delete the most used index items. For more information, see "onstat -C" on page 14-10.
PLOG_OVERFLOW_PATH	"PLOG_OVERFLOW_PATH" on page 1-64.

onstat Enhancements

Version 9.4 included the following enhancements to the **onstat** utility.

New Features	Reference
onstat -g env	This new **onstat -g** option displays the settings for all environment variables know to the server. For more information, see "The onstat -g env Option" on page 14-31.
onstat -g ses	The **onstat -g ses** option has been enhanced to provide additional session information used to diagnose shared memory dumps. For more information, see "The onstat -g ses Option" on page 14-67.
onstat -g sql	For more information, see "The onstat -g sql Option" on page 14-71.

Features From Dynamic Server, Version 9.3

The following tables provide information about features that were added to IBM Informix Dynamic Server, Version 9.3.

DataBlade API Enhancements

Version 9.3 included the following enhancements to the DataBlade API.

New Features	Reference
New PER_STMT_EXEC and PER_STMT_PREP memory durations	**onstat -g mem** in "onstat -g Monitoring Options" on page 14-16

Database Server Usability Enhancements

Version 9.3 included new features that make the database server easier to install, use, and manage.

New Features	Reference
Ability to display the maximum number of connections	Maximum server connections *number* in Appendix E, “Error Messages,” on page E-1
Microsoft Transaction Server/XA support	“onstat -x” on page 14-97
Two modifiable shell scripts, **alarmprogram.sh** and **alarmprogram.bat**, to handle event alarms	Appendix C, “Event Alarms,” on page C-1
Several new configuration parameters: • DEF_TABLE_LOCKMODE • DYNAMIC_LOGS • CDR_SERIAL • CDR_QDATA_SBSPACE • CDR_QHDR_DBSPACE • SBSPACETEMP	“DEF_TABLE_LOCKMODE” on page 1-27 “DYNAMIC_LOGS” on page 1-37 “Enterprise Replication Configuration Parameters” on page 1-38 “SBSPACETEMP” on page 1-69
Changed output for these commands: • **onstat -l** (displays temporary logs) • **onstat -x** (displays current log position)	“onstat -l” on page 14-82 “onstat -x” on page 14-97

Extensibility Enhancements

Version 9.3 included the following improvements in the area of extensibility.

Sbspace enhancements	Reference
Temporary sbspaces and smart large objects	“Create an Sbspace or Temporary Sbspace” on page 13-9
Improved space allocation of user data and metadata in sbspaces	“Sbspace Structure” on page 3-23 “Sbspace Metadata Messages” on page E-28

Performance Enhancements

Version 9.3 included many new features to help you monitor and improve the performance of your database.

New Features	Reference
The **onstat -g stm** option	**onstat -g stm** on page 14-16
Dynamic addition of logical logs	“DYNAMIC_LOGS” on page 1-37 “Add a Logical-Log File” on page 12-2 “Dynamic Log Messages” on page E-27

SQL Enhancements

Version 9.3 included several new SQL statements to ease migration from non-IBM Informix databases to Dynamic Server, Version 9.3.

New Features	Reference
Configurable default lock modes	“DEF_TABLE_LOCKMODE” on page 1-27

Other Significant Changes in Version 9.3

The following lists significant changes to the *IBM Informix Administrator's Reference*.

Changes to the Manual	Reference
Use the VPCLASS configuration parameter instead of the AFF_NPROCS, AFF_SPROC, NOAGE, NUMAIOVPS, and NUMCPUVPS configuration parameters.	Appendix D, "Discontinued Configuration Parameters," on page D-1.
The conversion and reversion error messages are now in this manual.	"Conversion/Reversion Messages" on page E-21

Features From Dynamic Server, Version 9.21

These features were introduced in IBM Informix Dynamic Server, Version 9.21.

Features	Reference
SQL statement cache enhancements: • new configuration parameters • **onstat -g ssc** option • **onstat -g ssc all** option • **onstat -g ssc pool** option • **onmode -W STMT_CACHE_HITS** • **onmode -W STMT_CACHE_NOLIMIT** • **onmode -W STMT_CACHE_SIZE**	"STMT_CACHE_HITS" on page 1-77 "STMT_CACHE_NOLIMIT" on page 1-77 "STMT_CACHE_NUMPOOL" on page 1-78 "onstat -g Monitoring Options" on page 14-16 "Change Settings for the SQL Statement Cache" on page 10-18
Java features: Drop JVPs	"Add or Remove Virtual Processors" on page 10-12

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other volumes in the documentation set.

The following conventions are discussed:

- Typographical conventions
- Other conventions
- Syntax diagrams
- Command-line conventions
- Example code conventions

Typographical Conventions

This manual uses the following conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

Convention	Meaning
KEYWORD	Keywords of SQL, SPL, and some other programming languages appear in uppercase letters in a serif font.

Convention	Meaning
italics	Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics.
boldface	Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface.
`monospace`	Information that the product displays and information that you enter appear in a monospace typeface.
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
>	This symbol indicates a menu item. For example, "Choose **Tools > Options**" means choose the **Options** item from the **Tools** menu.

Tip: When you are instructed to "enter" characters or to "execute" a command, immediately press RETURN after the entry. When you are instructed to "type" the text or to "press" other keys, no RETURN is required.

Feature, Product, and Platform Markup

Feature, product, and platform markup identifies paragraphs that contain feature-specific, product-specific, or platform-specific information. Some examples of this markup follow:

Dynamic Server

Identifies information that is specific to IBM Informix Dynamic Server

End of Dynamic Server

Windows Only

Identifies information that is specific to the Windows environment

End of Windows Only

This markup can apply to one or more paragraphs within a section. When an entire section applies to a particular product or platform, this is noted as part of the heading text, for example:

Table Sorting (Linux)

Syntax Diagrams

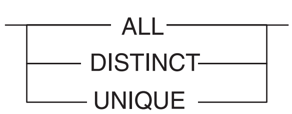
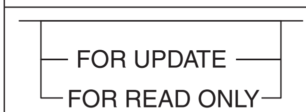
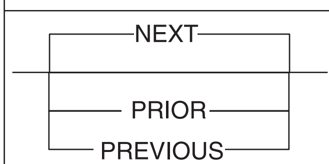
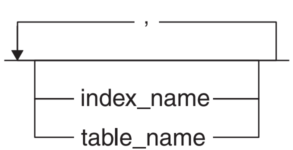
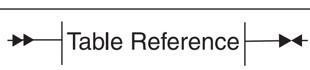
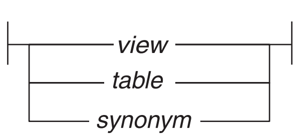
This guide uses syntax diagrams built with the following components to describe the syntax for statements and all commands other than system-level commands.

Syntax diagrams depicting SQL and command-line statements have changed in the following ways:

- The symbols at the beginning and end of statements are double arrows.
- The symbols at the beginning and end of syntax segment diagrams are vertical lines.
- How many times a loop can be repeated is explained in a diagram footnote, whose marker appears above the path that is describes.

- Syntax statements that are longer than one line continue on the next line.
- Product or condition-specific paths are explained in diagram footnotes, whose markers appear above the path that they describe.
- Cross-references to the descriptions of other syntax segments appear as diagram footnotes, whose markers immediately follow the name of the segment that they reference.

The following table describes syntax diagram components.

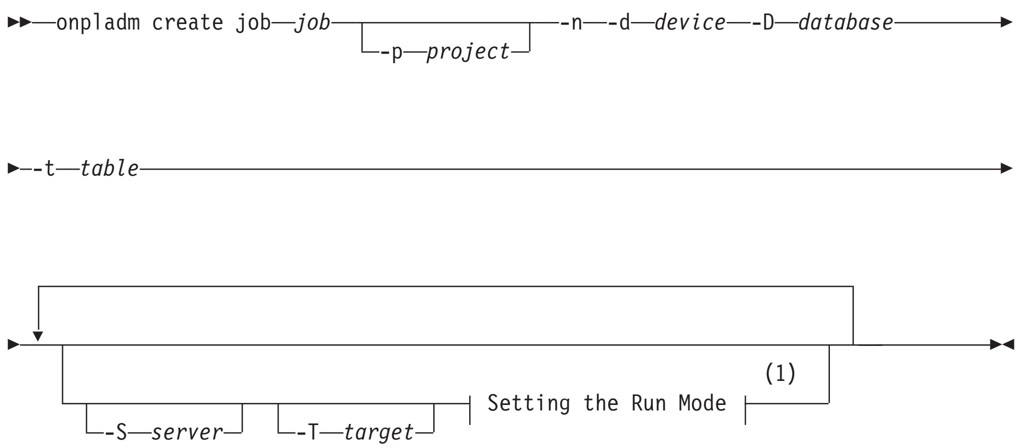
<table>
<tr><th>Component represented in PDF</th><th>Component represented in HTML</th><th>Meaning</th></tr>
<tr><td></td><td><pre>>>----------------------</pre></td><td>Statement begins.</td></tr>
<tr><td></td><td><pre>----------------------></pre></td><td>Statement continues on next line.</td></tr>
<tr><td></td><td><pre>>-----------------------</pre></td><td>Statement continues from previous line.</td></tr>
<tr><td></td><td><pre>-----------------------><</pre></td><td>Statement ends.</td></tr>
<tr><td></td><td><pre>--------SELECT----------</pre></td><td>Required item.</td></tr>
<tr><td></td><td><pre>--+----------------+---
  '------LOCAL------'</pre></td><td>Optional item.</td></tr>
<tr><td></td><td><pre>---+-----ALL-------+---
   +--DISTINCT-----+
   '---UNIQUE------'</pre></td><td>Required item with choice. One and only one item must be present.</td></tr>
<tr><td></td><td><pre>---+-----------------+---
    +--FOR UPDATE-----+
    '--FOR READ ONLY--'</pre></td><td>Optional items with choice are shown below the main line, one of which you might specify.</td></tr>
<tr><td></td><td><pre>   .---NEXT---------.
----+----------------+---
    +---PRIOR--------+
    '---PREVIOUS-----'</pre></td><td>The values below the main line are optional, one of which you might specify. If you do not specify an item, the value above the line will be used as the default.</td></tr>
<tr><td></td><td><pre> .-------,-----------.
 V |
---+-----------------+---
    +---index_name---+
    '---table_name---'</pre></td><td>Optional items. Several items are allowed; a comma must precede each repetition.</td></tr>
<tr><td></td><td><pre>>>-| Table Reference |-><</pre></td><td>Reference to a syntax segment.</td></tr>
<tr><td>Table Reference</td><td><pre>Table Reference

|--+-----view--------+--|
   +------table------+
   '----synonym------'</pre></td><td>Syntax segment.</td></tr>
</table>

How to Read a Command-Line Syntax Diagram

The following command-line syntax diagram uses some of the elements listed in the table in the previous section.

Creating a No-Conversion Job

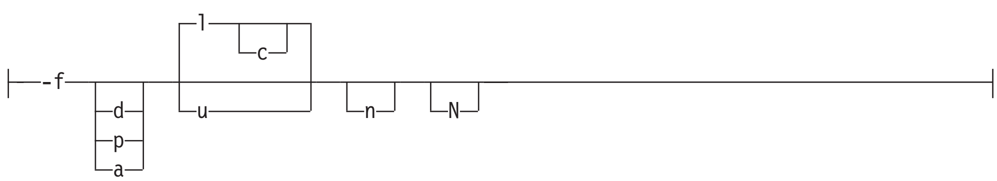


Notes:

1 See page Z-1

The second line in this diagram has a segment named "Setting the Run Mode," which according to the diagram footnote, is on page Z-1. If this was an actual cross-reference, you would find this segment in on the first page of Appendix Z. Instead, this segment is shown in the following segment diagram. Notice that the diagram uses segment start and end components.

Setting the Run Mode:



To see how to construct a command correctly, start at the top left of the main diagram. Follow the diagram to the right, including the elements that you want. The elements in this diagram are case sensitive because the illustrates utility syntax. Other types of syntax, such as SQL, are not case sensitive.

The Creating a No-Conversion Job diagram illustrates the following steps:

1. Type **onpladm create job** and then the name of the job.
2. Optionally, type **-p** and then the name of the project.
3. Type the following required elements:
   - **-n**
   - **-d** and the name of the device
   - **-D** and the name of the database
   - **-t** and the name of the table

4. Optionally, you can choose one or more of the following elements and repeat them an arbitrary number of times:
   - **-S** and the server name
   - **-T** and the target server name
   - The run mode. To set the run mode, follow the Setting the Run Mode segment diagram to type **-f**, optionally type **d**, **p**, or **a**, and then optionally type **l** or **u**.
5. Follow the diagram to the terminator.

Your diagram is complete.

Keywords and Punctuation

Keywords are words reserved for statements and all commands except system-level commands. When a keyword appears in a syntax diagram, it is shown in uppercase letters. When you use a keyword in a command, you can write it in uppercase or lowercase letters, but you must spell the keyword exactly as it appears in the syntax diagram.

You must also use any punctuation in your statements and commands exactly as shown in the syntax diagrams.

Identifiers and Names

Variables serve as placeholders for identifiers and names in the syntax diagrams and examples. You can replace a variable with an arbitrary name, identifier, or literal, depending on the context. Variables are also used to represent complex syntax elements that are expanded in additional syntax diagrams. When a variable appears in a syntax diagram, an example, or text, it is shown in *lowercase italic*.

The following syntax diagram uses variables to illustrate the general form of a simple SELECT statement.

```
►►──SELECT──column_name──FROM──table_name─────────────────────────────────────►◄
```

When you write a SELECT statement of this form, you replace the variables *column_name* and *table_name* with the name of a specific column and table.

Example Code Conventions

Examples of SQL code occur throughout this manual. Except as noted, the code is not specific to any single IBM Informix application development tool.

If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
   WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using DB–Access, you must delimit multiple statements with semicolons. If you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement.

Tip: Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the manual for your product.

Additional Documentation

For additional information, refer to the following types of documentation:

- Installation guides
- Online notes
- Informix error messages
- Manuals
- Online help

IBM Informix Information Center

The Informix Dynamic Server Information Center integrates the entire IBM Informix Dynamic Server 10.0 and IBM Informix Client SDK (CSDK) 2.90 documentation sets in both HTML and PDF formats. The Information Center provides full text search, a master index, logical categories, easy navigation, and links to troubleshooting and support files.

The IBM Informix Information Center site is located at http://publib.boulder.ibm.com/infocenter/ids9help/index.jsp.

Installation Guides

Installation guides are located in the **/doc** directory of the product CD or in the **/doc** directory of the product's compressed file if you downloaded it from the IBM Web site. Alternatively, you can obtain installation guides from the IBM Informix Online Documentation site at http://www.ibm.com/software/data/informix/pubs/library/ or the IBM Informix Information Center at http://publib.boulder.ibm.com/infocenter/ids9help/index.jsp.

Online Notes

The following sections describe the online files that supplement the information in this manual. Please examine these files before you begin using your IBM Informix product. They contain vital information about application and performance issues.

Online File	Description	Format
TOC Notes	The TOC (Table of Contents) notes file provides a comprehensive directory of hyperlinks to the release notes, the fixed and known defects file, and all the documentation notes files for individual manual titles.	HTML
Documentation Notes	The documentation notes file for each manual contains important information and corrections that supplement the information in the manual or information that was modified since publication.	HTML, text
Release Notes	The release notes file describes feature differences from earlier versions of IBM Informix products and how these differences might affect current products. For some products, this file also contains information about any known problems and their workarounds.	HTML, text
Machine Notes	(Non-Windows platforms only) The machine notes file describes any platform-specific actions that you must take to configure and use IBM Informix products on your computer.	text
Fixed and Known Defects File	This text file lists issues that have been identified with the current version. It also lists customer-reported defects that have been fixed in both the current version and in previous versions.	text

Locating Online Notes

Online notes are available from the IBM Informix Online Documentation site at http://www.ibm.com/software/data/informix/pubs/library/ and in the IBM Informix Information Center at http://publib.boulder.ibm.com/infocenter/ids9help/index.jsp. Additionally you can locate these files before or after installation as described below.

Before Installation

All online notes are located in the **/doc** directory of the product CD. The easiest way to access the documentation notes, the release notes, and the fixed and known defects file is through the hyperlinks from the TOC notes file.

The machine notes file and the fixed and known defects file are only provided in text format.

After Installation

On UNIX platforms in the default locale, the documentation notes, release notes, and machine notes files appear under the **$INFORMIXDIR/release/en_us/0333** directory.

Dynamic Server

On Windows the documentation and release notes files appear in the **Informix** folder. To display this folder, choose **Start > Programs > IBM** *product name version* **> Documentation Notes** or **Release Notes** from the taskbar.

Machine notes do not apply to Windows platforms.

End of Dynamic Server

Online Notes Filenames

Online notes have the following file formats:

Online File	File Format	Examples
TOC Notes	*prod_os*_toc_*version*.html	**ids_win_toc_10.0.html**
Documentation Notes	*prod_bookname*_docnotes_*version*.html/txt	**ids_hpl_docnotes_10.0.html**
Release Notes	*prod_os*_relnotes_*version*.html/txt	**ids_unix_relnotes_10.0.txt**
Machine Notes	*prod*_machine_notes_*version*.txt	**ids_machine_notes_10.0.txt**
Fixed and Known Defects File	*prod*_defects_*version*.txt ids_win_fixed_and_known _defects_*version*.txt	**ids_defects_10.0.txt** **client_defects_2.90.txt** **ids_win_fixed_and_known _defects_10.0.txt**

Informix Error Messages

This file is a comprehensive index of error messages and their corrective actions for the Informix products and version numbers.

On UNIX platforms, use the **finderr** command to read the error messages and their corrective actions.

Dynamic Server

On Windows, use the Informix Error Messages utility to read error messages and their corrective actions. To display this utility, choose **Start > Programs > IBM *product name version* > Informix Error Messages** from the taskbar.

End of Dynamic Server

You can also access these files from the IBM Informix Online Documentation site at http://www.ibm.com/software/data/informix/pubs/library/ or in the IBM Informix Information Center at http://publib.boulder.ibm.com/infocenter/ids9help/index.jsp.

Manuals

Online Manuals

A CD that contains your manuals in electronic format is provided with your IBM Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print online manuals, see the installation insert that accompanies your CD. You can also obtain the same online manuals from the IBM Informix Online Documentation site at http://www.ibm.com/software/data/informix/pubs/library/ or in the IBM Informix Information Center at http://publib.boulder.ibm.com/infocenter/ids9help/index.jsp.

Printed Manuals

To order hardcopy manuals, contact your sales representative or visit the IBM Publications Center Web site at http://www.ibm.com/software/howtobuy/data.html.

Online Help

IBM Informix online help, provided with each graphical user interface (GUI), displays information about those interfaces and the functions that they perform. Use the help facilities that each GUI provides to display the online help.

Accessibility

IBM is committed to making our documentation accessible to persons with disabilities. Our books are available in HTML format so that they can be accessed with assistive technology such as screen reader software. The syntax diagrams in our manuals are available in dotted decimal format, which is an accessible format that is available only if you are using a screen reader. For more information about the dotted decimal format, see the Accessibility appendix.

IBM Informix Dynamic Server Version 10.0 and CSDK Version 2.90 Documentation Set

The following tables list the manuals that are part of the IBM Informix Dynamic Server, Version 10.0 and the CSDK Version 2.90, documentation set. PDF and HTML versions of these manuals are available at http://www.ibm.com/software/data/informix/pubs/library/ or in the IBM Informix Information Center at http://publib.boulder.ibm.com/infocenter/ids9help/index.jsp. You can order hardcopy versions of these manuals from the IBM Publications Center at http://www.ibm.com/software/howtobuy/data.html.

Table 1. Database Server Manuals

Manual	Subject
Administrator's Guide	Understanding, configuring, and administering your database server.
Administrator's Reference	Reference material for Informix Dynamic Server, such as the syntax of database server utilities **onmode** and **onstat**, and descriptions of configuration parameters, the **sysmaster** tables, and logical-log records.
Backup and Restore Guide	The concepts and methods you need to understand when you use the **ON-Bar** and **ontape** utilities to back up and restore data.
Built-In DataBlade Modules User's Guide	Using the following DataBlade modules that are included with Dynamic Server: • MQ DataBlade module, to allow IBM Informix database applications to communicate with other MQSeries applications. • Large Object Locator, a foundation DataBlade module that can be used by other modules that create or store large-object data.
DB-Access User's Guide	Using the **DB-Access** utility to access, modify, and retrieve data from Informix databases.
DataBlade API Function Reference	The DataBlade API functions and the subset of ESQL/C functions that the DataBlade API supports. You can use the DataBlade API to develop client LIBMI applications and C user-defined routines that access data in Informix databases.
DataBlade API Programmer's Guide	The DataBlade API, which is the C-language application-programming interface provided with Dynamic Server. You use the DataBlade API to develop client and server applications that access data stored in Informix databases.

Table 1. Database Server Manuals (continued)

Manual	Subject
Database Design and Implementation Guide	Designing, implementing, and managing your Informix databases.
Enterprise Replication Guide	How to design, implement, and manage an Enterprise Replication system to replicate data between multiple database servers.
Error Messages file	Causes and solutions for numbered error messages you might receive when you work with IBM Informix products.
Getting Started Guide	Describes the products bundled with IBM Informix Dynamic Server and interoperability with other IBM products. Summarizes important features of Dynamic Server and the new features for each version.
Guide to SQL: Reference	Information about Informix databases, data types, system catalog tables, environment variables, and the stores_demo demonstration database.
Guide to SQL: Syntax	Detailed descriptions of the syntax for all Informix SQL and SPL statements.
Guide to SQL: Tutorial	A tutorial on SQL, as implemented by Informix products, that describes the basic ideas and terms that are used when you work with a relational database.
High-Performance Loader User's Guide	Accessing and using the High-Performance Loader (HPL), to load and unload large quantities of data to and from Informix databases.
Installation Guide for Microsoft Windows	Instructions for installing IBM Informix Dynamic Server on Windows.
Installation Guide for UNIX and Linux	Instructions for installing IBM Informix Dynamic Server on UNIX and Linux.
J/Foundation Developer's Guide	Writing user-defined routines (UDRs) in the Java programming language for Informix Dynamic Server with J/Foundation.
Migration Guide	Conversion to and reversion from the latest versions of Informix database servers. Migration between different Informix database servers.
Optical Subsystem Guide	The Optical Subsystem, a utility that supports the storage of BYTE and TEXT data on optical disk.
Performance Guide	Configuring and operating IBM Informix Dynamic Server to achieve optimum performance.
R-Tree Index User's Guide	Creating R-tree indexes on appropriate data types, creating new operator classes that use the R-tree access method, and managing databases that use the R-tree secondary access method.
SNMP Subagent Guide	The IBM Informix subagent that allows a Simple Network Management Protocol (SNMP) network manager to monitor the status of Informix servers.
Storage Manager Administrator's Guide	Informix Storage Manager (ISM), which manages storage devices and media for your Informix database server.
Trusted Facility Guide	The secure-auditing capabilities of Dynamic Server, including the creation and maintenance of audit logs.
User-Defined Routines and Data Types Developer's Guide	How to define new data types and enable user-defined routines (UDRs) to extend IBM Informix Dynamic Server.
Virtual-Index Interface Programmer's Guide	Creating a secondary access method (index) with the Virtual-Index Interface (VII) to extend the built-in indexing schemes of IBM Informix Dynamic Server. Typically used with a DataBlade module.
Virtual-Table Interface Programmer's Guide	Creating a primary access method with the Virtual-Table Interface (VTI) so that users have a single SQL interface to Informix tables and to data that does not conform to the storage scheme of Informix Dynamic Server.

Table 2. Client/Connectivity Manuals

Manual	Subject
Client Products Installation Guide	Installing IBM Informix Client Software Developer's Kit (Client SDK) and IBM Informix Connect on computers that use UNIX, Linux, and Windows.
Embedded SQLJ User's Guide	Using IBM Informix Embedded SQLJ to embed SQL statements in Java programs.
ESQL/C Programmer's Manual	The IBM Informix implementation of embedded SQL for C.
GLS User's Guide	The Global Language Support (GLS) feature, which allows IBM Informix APIs and database servers to handle different languages, cultural conventions, and code sets.
JDBC Driver Programmer's Guide	Installing and using Informix JDBC Driver to connect to an Informix database from within a Java application or applet.
.NET Provider Reference Guide	Using Informix .NET Provider to enable .NET client applications to access and manipulate data in Informix databases.
ODBC Driver Programmer's Manual	Using the Informix ODBC Driver API to access an Informix database and interact with the Informix database server.
OLE DB Provider Programmer's Guide	Installing and configuring Informix OLE DB Provider to enable client applications, such as ActiveX Data Object (ADO) applications and Web pages, to access data on an Informix server.
Object Interface for C++ Programmer's Guide	The architecture of the C++ object interface and a complete class reference.

Table 3. DataBlade Developer's Kit Manuals

Manual	Subject
DataBlade Developer's Kit User's Guide	Developing and packaging DataBlade modules using BladeSmith and BladePack.
DataBlade Module Development Overview	Basic orientation for developing DataBlade modules. Includes an example illustrating the development of a DataBlade module.
DataBlade Module Installation and Registration Guide	Installing DataBlade modules and using BladeManager to manage DataBlade modules in Informix databases.

Compliance with Industry Standards

The American National Standards Institute (ANSI) and the International Organization of Standardization (ISO) have jointly established a set of industry standards for the Structured Query Language (SQL). IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

IBM Welcomes Your Comments

We want to know about any corrections or clarifications that you would find useful in our manuals, which will help us improve future versions. Include the following information:

- The name and version of the manual that you are using
- Section and page number
- Your suggestions about the manual

Send your comments to us at the following email address:

docinf@us.ibm.com

This email address is reserved for reporting errors and omissions in our documentation. For immediate help with a technical problem, contact IBM Technical Support. For instructions, see the IBM Informix Technical Support website at http://www-306.ibm.com/software/data/informix/support/contact.html.

We appreciate your suggestions.

Part 1. Configuring and Monitoring Dynamic Server

Chapter 1. Configuration Parameters

In This Chapter

This chapter describes the **ONCONFIG** file conventions, lists the configuration parameters in the **ONCONFIG** file, and provides a short discussion of each parameter.

ONCONFIG File Conventions

The **ONCONFIG** environment variable specifies the file that contains the configuration parameters. This file is also called the ***ONCONFIG file***. The database server uses the **ONCONFIG** file during initialization.

Format of ONCONFIG File

In the **ONCONFIG** file, each parameter is on a separate line. The file can also contain blank lines and comment lines that start with a # symbol. The following line shows the syntax for a parameter line:

```
PARAMETER_NAME    parameter_value    #comment
```

Parameters and their values in the **ONCONFIG** file are case sensitive. The parameter names are always uppercase. If the value entry is described with uppercase letters, you must use uppercase (for example, the CPU value of the NETTYPE parameter). You must put white space (tabs, spaces, or both) between the parameter name, parameter value, and optional comment. Do not use any tabs or spaces within a parameter value.

Restriction: The maximum line limit of the ONCONFIG file is 512 bytes. Lines that exceed this limit are truncated and might cause configuration problems.

ONCONFIG File Templates

The database server provides a template for a configuration file that contains initial values for many of the ONCONFIG parameters.

IBM Informix Dynamic Server provides **onconfig.std** as a template configuration file that you can copy and tailor to your specific configuration.

If you omit a parameter value in your copy of the configuration file, the database server either uses default values in **onconfig.std** or calculates values based on other parameter values. For information on the order of files in which the database server looks for configuration values during initialization, refer to the chapter on initializing the database server in the *IBM Informix Administrator's Guide*.

Warning: Do not modify or delete **onconfig.std**, which is a template and not a functional configuration.

The following table lists the locations of the **ONCONFIG** and **onconfig.std** files.

Operating System	ONCONFIG File	Template File
UNIX	**$INFORMIXDIR/etc/$ONCONFIG**	**$INFORMIXDIR/etc/onconfig.std**
Windows	**%INFORMIXDIR%\etc\%ONCONFIG%**	**%INFORMIXDIR%\etc\onconfig.std**

To prepare the ONCONFIG file:

1. Copy the **onconfig.std** template file.
2. Modify the *copy* of the template file.
3. Set the **ONCONFIG** environment variable to the name of the copy of the pertinent template file.

   If you do not set **ONCONFIG**, the default filename is **onconfig**.

For more details on why you might want to modify the default configuration parameters, refer to the chapter on configuring the database server in the *IBM Informix Administrator's Guide*.

Printing the onconfig.std File

Important: Print out a copy of the **onconfig.std** file to see the latest default values for the configuration parameters and recommended settings.

Specifying Hidden Configuration Parameters

A few of the configuration parameters, such as DYNAMIC_LOGS, are omitted from the **onconfig.std** file. It is recommended that you use the default values for these hidden parameters. If you want to change the value for a hidden parameter, add it to your ONCONFIG file.

Displaying ONCONFIG Settings

When the database server restarts, it reads the ONCONFIG file. To view the ONCONFIG settings, use one of the following tools:

- IBM Informix Server Administrator (ISA)
- **oncheck -pr**

  The information under PAGE_CONFIG lists the configuration parameter settings at restart. For more information, see "Display Reserved-Page Information with -pr and -pR" on page 6-15.
- **.infos.dbservername**

  If you set the **ONCONFIG** environment variable to the name of a different ONCONFIG file while the database server is online, the **.infos.dbservername** file contains the current settings. For more information, see ".infos.dbservername" on page A-6 and "The ONCONFIG File" on page A-7.

For more information about the **ONCONFIG** environment variable, see the *IBM Informix Guide to SQL: Reference*.

Summary of Configuration Parameters

This section provides the following information:

- A list of each configuration parameter with database server compatibility
- A description of the attributes listed for each configuration parameter

The configuration parameters and database server compatibility are as follows. For information on the discontinued configuration parameters, see Appendix D. If the configuration parameter has a related environment variable, it is listed in the following table.

Configuration Parameter	Related Environment Variable	Reference
AC_DEBUG	**AC_CONFIG**	page 1-11
AC_IXBAR	**AC_CONFIG**	page 1-11
AC_LTAPEBLOCK	**AC_CONFIG**	page 1-11
AC_LTAPEDEV	**AC_CONFIG**	page 1-11
AC_MSGPATH	**AC_CONFIG**	page 1-11
AC_SCHEMA	**AC_CONFIG**	page 1-11
AC_STORAGE	**AC_CONFIG**	page 1-11
AC_TAPEBLOCK	**AC_CONFIG**	page 1-11
AC_TAPEDEV	**AC_CONFIG**	page 1-11

Configuration Parameter	Related Environment Variable	Reference
AC_TIMEOUT	**AC_CONFIG**	page 1-11
AC_VERBOSE	**AC_CONFIG**	page 1-11
ADTERR		page 1-11
ADTMODE		page 1-11
ADTPATH		page 1-11
ADTSIZE		page 1-11
AFCRASH		page 1-43
AFF_NPROCS		page D-1
AFF_SPROC		page D-2
ALARMPROGRAM		page 1-12
ALLOW_NEWLINE		page 1-12
ALRM_ALL_EVENTS		page 1-13
BAR_ACT_LOG		page 1-58
BAR_BSALIB_PATH		page 1-58
BAR_DEBUG		page 1-58
BAR_DEBUG_LOG		page 1-58
BAR_HISTORY		page 1-58
BAR_MAX_BACKUP		page 1-58
BAR_NB_XPORT_COUNT		page 1-58
BAR_PROGRESS_FREQ		page 1-58
BAR_RETRY		page 1-58
BAR_XFER_BUF_SIZE		page 1-58
BLOCKTIMEOUT		page 1-13
BTSCANNER		page 1-13
BUFFERPOOL		page 1-14
CDR_DBSPACE		page 1-38
CDR_DSLOCKWAIT		page 1-38
CDR_ENV		page 1-38
CDR_EVALTHREADS		page 1-38
CDR_MAX_DYNAMIC_LOGS		page 1-38
CDR_NIFCOMPRESS		page 1-38
CDR_QDATA_SBSPACE		page 1-38
CDR_QHDR_DBSPACE		page 1-38
CDR_QUEUEMEM		page 1-38
CDR_SERIAL		page 1-38
CDR_SUPPRESS_ATSRISWARN		page 1-38
CKPTINTVL		page 1-19
CLEANERS		page 1-19
CONSOLE		page 1-20
DATASKIP		page 1-20
4 DB_LIBRARY_PATH		page 1-21

Configuration Parameter	Related Environment Variable	Reference
DBSERVERALIASES		page 1-22
DBSERVERNAME	**INFORMIXSERVER**	page 1-22
DBSPACETEMP	**DBSPACETEMP**	page 1-23
DD_HASHMAX		page 1-25
DD_HASHSIZE		page 1-26
DEADLOCK_TIMEOUT		page 1-26
DEF_TABLE_LOCKMODE	**IFX_DEF_TABLE_LOCKMODE**	page 1-27
DIRECTIVES	**IFX_DIRECTIVES**	page 1-27
DISABLE_B162428_XA_FIX	**IFX_XASTDCOMPLIANCE _XAEND**	page 1-28
DRIDXAUTO		page 1-28
DRAUTO		page 1-29
DRINTERVAL		page 1-29
DRLOSTFOUND		page 1-30
DRTIMEOUT		page 1-30
DS_HASHSIZE		page 1-31
DS_MAX_QUERIES		page 1-31
DS_MAX_SCANS		page 1-32
DS_NONPDQ_QUERY_MEM		page 1-33
DS_POOLSIZE		page 1-33
DS_TOTAL_MEMORY		page 1-34
DUMPCNT		page 1-35
DUMPCORE		page 1-35
DUMPDIR		page 1-36
DUMPGCORE		page 1-36
DUMPSHMEM		page 1-37
DYNAMIC_LOGS		page 1-37
ENCRYPT_CDR		page 1-37
ENCRYPT_CIPHERS		page 1-37
ENCRIPT_MAC		page 1-37
ENCRYPT_MACFILE		page 1-37
ENCRYPT_SWITCH		page 1-37
EXT_DIRECTIVES	**IFX_EXTDIRECTIVES**	page 1-39
EXTSHMADD		page 1-40
FAST_RESTART_CKPT_ FUZZYLOG		page 1-40
FAST_RESTART_PHYSLOG		page 1-40
FILLFACTOR		page 1-41
HETERO_COMMIT		page 1-42
IFX_EXTEND_ROLE		page 1-42
IMCLOG		page 1-51

Configuration Parameter	Related Environment Variable	Reference
IMCTRANSPORTS		page 1-51
IMCWORKERDELAY		page 1-51
IMCWORKERTHREADS		page 1-51
ISM_DATA_POOL		page 1-42
ISM_LOG_POOL		page 1-42
JDKVERSION		page 1-43
JVMTHREAD		page 1-43
JVPCLASSPATH		page 1-43
JVPDEBUG		page 1-43
JVPHOME		page 1-43
JVPJAVAHOME		page 1-43
JVPJAVALIB		page 1-43
JVPJAVAVM		page 1-43
JVPLOGFILE		page 1-43
JVPPROPFILE		page 1-43
LISTEN_TIMEOUT		page 1-43
LOCKS		page 1-44
LOGBUFF		page 1-44
LOGFILES		page 1-45
LOGSIZE		page 1-45
LTAPEBLK		page 1-46
LTAPEDEV		page 1-47
LTAPESIZE		page 1-48
LTXEHWM		page 1-48
LTXHWM		page 1-49
MAX_INCOMPLETE_ CONNECTIONS		page 1-50
MAX_PDQPRIORITY		page 1-50
MIRROR		page 1-51
MIRROROFFSET		page 1-52
MIRRORPATH		page 1-52
MSGPATH		page 1-53
MULTIPROCESSOR		page 1-53
NETTYPE		page 1-54
NOAGE		page D-5
NUMAIOVPS		page D-6
NUMCPUVPS		page D-6
OFF_RECVRY_THREADS		page 1-57
ON_RECVRY_THREADS		page 1-57
ONDBSPACEDOWN		page 1-58
ONLIDX_MAXMEM		page 1-59

Configuration Parameter	Related Environment Variable	Reference
OPCACHEMAX	**INFORMIXOPCACHE**	page 1-59
OPTCOMPIND	**OPTCOMPIND**	page 1-60
OPT_GOAL	**OPT_GOAL**	page 1-61
PC_HASHSIZE		page 1-62
PC_POOLSIZE		page 1-62
PHYSBUFF		page 1-62
PHYSDBS		page 1-63
PHYSFILE		page 1-63
RA_PAGES		page 1-64
RA_THRESHOLD		page 1-64
RESIDENT		page 1-65
RESTARTABLE_RESTORE		page 1-66
ROOTNAME		page 1-67
ROOTOFFSET		page 1-67
ROOTPATH		page 1-67
ROOTSIZE		page 1-68
SBSPACENAME		page 1-68
SBSPACETEMP		page 1-69
SECURESTATG		page 1-70
SECURITY_LOCALCONNECTION		page 1-70
SERVERNUM		page 1-71
SHMADD		page 1-71
SHMBASE		page 1-72
SHMTOTAL		page 1-72
SHMVIRTSIZE		page 1-73
SINGLE_CPU_VP		page 1-74
STACKSIZE	**INFORMIXSTACKSIZE**	page 1-75
STAGEBLOB		page 1-76
STMT_CACHE	**STMT_CACHE**	page 1-76
STMT_CACHE_HITS		page 1-77
STMT_CACHE_NOLIMIT		page 1-77
STMT_CACHE_NUMPOOL		page 1-78
STMT_CACHE_SIZE		page 1-79
SYSALARMPROGRAM		page 1-79
SYSSBSPACENAME		page 1-80
TAPEBLK		page 1-81
TAPEDEV		page 1-81
TAPESIZE		page 1-82
TBLSPACE_STATS		page 1-83
TBLTBLFIRST		page 1-83
TBLTBLNEXT		page 1-84

Configuration Parameter	Related Environment Variable	Reference
TXTIMEOUT		page 1-84
USEOSTIME		page 1-84
VPCLASS		page 1-85

Parameter Attributes

This chapter describes one or more of the following attributes (if relevant) for each parameter.

Attribute	Description
onconfig.std value	The default value that appears in the **onconfig.std** file. The database server uses these default values for all configurations.
if not present	The value that the database server supplies if the parameter is missing from your ONCONFIG file. If this value is present in **onconfig.std**, the database server uses the **onconfig.std** value. If this value is not present in **onconfig.std**, the database server calculates the value based on other values in **onconfig.std**.
units	The units in which the parameter is expressed
separators	The separators that can be used when the parameter value has several parts Do *not* use white space within a parameter value.
range of values	The valid values for this parameter
takes effect	The time at which a change to the value of the parameter affects the operation of the database server. *Disk is initialized* means to reinitialize the database server.
utilities	The database server utilities that you can use to change the value of the parameter
refer to	Cross-reference to further discussion

Using a Utility to Change a Parameter Value

Use one of these utilities to change the value of a configuration parameter. The *utilities* section for each configuration parameter lists the specific utilities to use.

Tool	Description
ON–Monitor (UNIX)	You can use ON–Monitor to change certain parameter values. In ON–Monitor, some of the responses are Y/N (yes/no). When those responses are recorded in the ONCONFIG file, Y becomes 1, and N becomes 0.
ISA	To use IBM Informix Server Administrator (ISA) to change parameter values, select **Configuration > ONCONFIG**.

Command-line utility	The *utilities* section lists one or more command-line utilities that you can use to change a parameter value.
Text editor	You can use a text editor to modify the ONCONFIG file.

Environment Variables

If you set the environment variable on the database server, it applies to all sessions. If you set the environment variable in the client environment, it applies to the current session and overrides the equivalent configuration parameter (if any). For a complete list of environment variables, and how to set them, see the *IBM Informix Guide to SQL: Reference*.

Archecker Configuration Parameters

The **ac_config.std** template contains the default **archecker** configuration parameters. Usually, you would not change these parameters. However, if you need to change these parameters, copy the **ac_config.std** template to the AC_CONFIG file. (The **AC_CONFIG** environment variable specifies the location of the AC_CONFIG file.) The **archecker** utility uses these parameters when it verifies a backup or performs a table-level restore. For information on these parameters, see the *IBM Informix Backup and Restore Guide*.

Configuration Parameter	Description
AC_DEBUG	Prints debugging messages in the **archecker** message log.
AC_IXBAR	Specifies the pathname to the IXBAR file.
AC_LTAPEBLOCK	Specifies the **ontape** block size for reading logical logs.
AC_LTAPEDEV	Specifies the local device name used by **ontape** for reading logical logs.
AC_MSGPATH	Specifies the location of the **archecker** message file.
AC_SCHEMA	Specifies the pathname to the **archecker schema** command.
AC_STORAGE	Specifies the location of the temporary files that **archecker** builds.
AC_TAPEBLOCK	Specifies the tape block size in kilobytes.
AC_TAPEDEV	Specifies the device name used by the **ontape** utility.
AC_TIMEOUT	Specifies the timeout value for ON-Bar and **archecker** processes if one of them exits prematurely.
AC_VERBOSE	Specifies either verbose or quiet mode for **archecker** messages.

ADTERR, ADTMODE, ADTPATH, and ADTSIZE (UNIX)

ADTERR, ADTMODE, ADTPATH, and ADTSIZE are configuration parameters for auditing. For information on these parameters, see the *IBM Informix Trusted Facility Guide*.

ALARMPROGRAM

onconfig.std *value*	On UNIX: **/usr/informix/etc/no_log.sh** On Windows: **%INFORMIXDIR%\etc\no_log.bat**
if not present	On UNIX: **/usr/informix/etc/no_log.sh** On Windows: **%INFORMIXDIR%\etc\no_log.bat**
range of values	*Full pathname*
takes effect	When the database server is shut down and restarted
refer to	The following material: • "Writing Your Own Alarm Script" on page C-1 • *IBM Informix Backup and Restore Guide*

Use the ALARMPROGRAM parameter to display event alarms. The following sample scripts are provided.

Script Name	Platform	Description
log_full.sh	UNIX	To back up logical logs automatically when the database server issues a log-full event alarm, set ALARMPROGRAM to **log_full.sh** or **log_full.bat**.
log_full.bat	Windows	
no_log.sh	UNIX	To disable automatic logical-log backups, set ALARMPROGRAM to **no_log.sh** or **no_log.bat**.
no_log.bat	Windows	
alarmprogram.sh	UNIX	Handles event alarms and controls logical-log backups. Modify **alarmprogram.sh** or **alarmprogram.bat** and set ALARMPROGRAM to the full pathname of **alarmprogram.sh** or **alarmprogram.bat**. See "Customizing the ALARMPROGRAM scripts" on page C-1.
alarmprogram.bat	Windows	

Important: Backup media should always be available for automatic log backups.

You can set the ALRM_ALL_EVENTS configuration parameter to specify whether ALARMPROGRAM runs for all events that are logged in the MSGPATH or only for specified noteworthy events (events greater than severity 1).

Instead of using the supplied scripts, you can write your own shell script, batch file, or binary program to execute events. Set ALARMPROGRAM to the full pathname of this file. The database server executes this script when noteworthy events occur. These events include database, table, index, or simple-large-object failure; all logs are full; internal subsystem failure; initialization failure; and long transactions. You can have the events noted in an email or pagermail message.

ALLOW_NEWLINE

onconfig.std *value*	`0`
range of values	`0` = Disallow the newline character in quoted strings for all sessions. `1` = Allow the newline character in quoted strings for all sessions.

takes effect	When the database server is shut down and restarted
refer to	The following material: • Quoted strings in the *IBM Informix Guide to SQL: Syntax* • Newline characters in quoted strings in the *IBM Informix ESQL/C Programmer's Manual*

You can specify that you want the database server to allow the newline character (**\n**) in a quoted string either for all sessions or for a specific session. A session is the duration of a client connection to the database server.

To allow or disallow newline characters in quoted strings for all sessions, set the ALLOW_NEWLINE parameter in the ONCONFIG file. To allow all remote sessions in a distributed query to support embedded newline characters, specify ALLOW_NEWLINE in their ONCONFIG files.

To allow or disallow a newline character in a quoted string for a particular session when ALLOW_NEWLINE is not set, you must execute the **ifx_allow_newline(boolean)** user-defined routine (UDR).

ALRM_ALL_EVENTS

onconfig.std *value*	0
takes effect	When the database server is shut down and restarted
range of values	0, 1

ALRM_ALL_EVENTS specifies whether ALARMPROGRAM runs for all events that are logged in the MSGPATH or only for noteworthy events. If ALRM_ALL_EVENTS is set to 1, it will trigger the ALARMPROGRAM and it will display all event alarms.

BLOCKTIMEOUT

onconfig.std *value*	3600
units	Seconds
takes effect	When the database server is shut down and restarted

BLOCKTIMEOUT specifies the number of seconds that a thread or database server will hang. After the timeout, the thread or database server will either continue processing or fail.

BTSCANNER

syntax

```
BTSCANNER [num=scanner_threads,][priority=low|high,]
[threshold=committed_deleted],[rangesize=size]
```

onconfig.std value

none

range of values

scanner_threads

The number of B-tree scanner threads to start. Must be an integer between `1` and `32`, inclusive. The default is `1`.

committed_deleted

The minimum number of committed deleted items an index must encounter before adding the index to the hot list as a candidate for cleaning. Must be an integer greater than `0`. The default is `500`.

takes effect

When the database server is initialized. You can adjust these B-tree scanner settings with the **onmode -C** command while the database server is online.

additional information

See "Control the B-tree Scanner" on page 10-7.

After all indexes above the threshold are cleaned, indexes below the threshold are added to the hot list. The default threshold is 500.

The `priority=high` option sets the priority of the B-tree scanner threads to equal that of normal users. The `priority=low` option sets the priority of the B-tree scanner threads lower than that of normal users. This command allows the B-tree scanner to consume only spare system resources, ensuring that the threads will not use the CPU cycles of normal users. The default priority is low.

BUFFERPOOL

onconfig.std

values

UNIX Only

```
BUFFERPOOL default,lrus=8,buffers=5000,lru_min_dirty=50,
lru_max_dirty=60
BUFFERPOOL size=2K,buffers=5000,lrus=8,lru_min_dirty=50,
lru_max_dirty=60
```

End of UNIX Only

Windows Only

```
BUFFERPOOL default,lrus=8,buffers=2000,lru_min_dirty=50,
lru_max_dirty=60
BUFFERPOOL size=4K,buffers=2000,lrus=8,lru_min_dirty=50,
lru_max_dirty=60
```

End of Windows Only

syntax

`BUFFERPOOL default,lrus=`*num_lrus*`,buffers=`*num_buffers*`,`
`lru_min_dirty=`*percent_min*`,lru_max_dirty=`*percent_max_dirty*
`BUFFERPOOL size=`*size*`K,buffers=`*num_buffers*`,`
`lrus=`*num_lrus*`,lru_min_dirty=`*percent_min*`,`
`lru_max_dirty=`*percent_max_dirty*

takes effect

When the database server is shut down and restarted

utilities

onparams -b (See "Add a New Buffer Pool" on page 12-4.)

onspaces (See "Specifying a Non-Default Page Size with the Same Size as the Buffer Pool" on page 13-6.
ON-Monitor (See Figure 11-7 on page 11-5.)

refer to "Create a Dbspace or Temporary Dbspace" on page 13-3
The *IBM Informix Dynamic Server Administrator's Guide*

The BUFFERPOOL configuration parameter specifies the default values for buffers and LRU queues in a buffer pool for both the default page size buffer pool and for any non-default pages size buffer pools.

Note: Information that was specified with the BUFFERS, LRUS, LRU_MAX_DIRTY, and LRU_MIN_DIRTY configuration parameters prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter.

The BUFFERPOOL configuration parameter consists of two lines in the **onconfig.std** file, as shown in this example for a UNIX platform:

```
BUFFERPOOL default,lrus=8,buffers=5000,lru_min_dirty=50,lru_max_dirty=60
BUFFERPOOL size=2K,buffers=5000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
```

The top line specifies the default values that are used if you create a dbspace with a page size that does not already have a corresponding buffer pool created at start up. The line below the default line specifies the database server's default values for a buffer pool, which are based on the database server's default page size. When you add a dbspace with a different page size with the **onspaces** utility or when you add a new buffer pool with the **onparams** utility, a new line is appended to the BUFFERPOOL configuration parameter in the ONCONFIG file. The page size for each buffer pool must be a multiple of the system's default page size. Below is an example of the BUFFERPOOL lines where a third line has been appended:

```
BUFFERPOOL default,lrus=8,buffers=5000,lru_min_dirty=50,lru_max_dirty=60
BUFFERPOOL size=2K,buffers=5000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
BUFFERPOOL size=6K,buffers=3000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
```

The order of precedence for the BUFFERPOOL configuration parameter settings is:

1. The BUFFERPOOL size line, for example:
   ```
   BUFFERPOOL size=2K,buffers=5000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
   ```
2. Any deprecated parameters in the ONCONFIG file:
   - BUFFERS
   - LRUS
   - LRU_MAX_DIRTY
   - LRU_MIN_DIRTY

   For more information about deprecated configuration parameters, see Appendix D, "Discontinued Configuration Parameters," on page D-1.
3. The BUFFERPOOL default line, for example:
   ```
   BUFFERPOOL default,lrus=8,buffers=5000,lru_min_dirty=50,lru_max_dirty=60
   ```
4. Database server defaults.

When you use **onspaces** to create a new dbspace with a new page size, the database server takes the values of **buffers**, **lrus**, **lru_min_dirty** and **lru_max_dirty** from BUFFERPOOL default line unless there already is a BUFFERPOOL entry for that page size.

You can use the **onparams** utility when the database server is in online, quiescent, or in single-user mode to add a new buffer pool with a different page size. There must be one buffer pool for each page size used by the dbspaces and all dbspaces using that page size must use the single buffer pool with that page size. When you use the **onparams** utility to add a buffer pool or when you add a dbspace with a different page size with the **onspaces** utility, the information you specify is automatically appended to the ONCONFIG file and new values are specified using the BUFFERPOOL keyword. You cannot change the values by editing the **onconfig.std** file. If you need to resize or delete an existing buffer pool, you must restart the database server and then run **onparams** again.

Buffer pools that are added while the database server is running go into virtual memory, not into resident memory. Only those buffer pool entries that are specified in the ONCONFIG file at startup go into resident memory, depending on the availability of the memory you are using.

The fields in the BUFFERPOOL lines are not case sensitive (so you can specify **lrus** or **Lrus** or **LRUS**) and the fields can appear in any order.

For more information on buffer pools, including information on resizing and deleting buffer pools, see *IBM Informix Dynamic Server Administrator's Guide*.

The following sections explain each of the fields in the BUFFERPOOL configuration parameter.

The lrus Field

onconfig.std *value*	`lrus=8`
syntax	`lrus=`*num_lrus*
units	Number of LRU queues
range of values	32-bit platforms: 1 through 128 64-bit platforms: 1 through 512

The **lrus** field specifies the number of LRU (least-recently-used) queues in the shared-memory buffer pool. You can tune the value of **lrus**, in combination with the **lru_min_dirty** and **lru_max_dirty** fields, to control how frequently the shared-memory buffers are flushed to disk.

Setting **lrus** too high might result in excessive page-cleaner activity.

The buffers Field

onconfig.std ***value***	UNIX: `buffers=5000` Windows: `buffers=2000`
syntax	`buffers=`*num_buffers*
units	Number of buffers. Each buffer is the size of the operating system page.

range of values	For 32-bit platform on UNIX:
	with page size equal to 2048 bytes:
	100 through 1,843,200 buffers
	`(1843200 = 1800 * 1024)`
	
	with page size equal to 4096 bytes:
	100 through 921,600 buffers
	`(921,600 = ((1800 * 1024)/4096) * 2048 )`

For 32-bit platform on Windows:
100 through 524,288 buffers (`524,288 = 512 * 1024`)

For 64-bit platforms: 100 through $2^{31}-1$ buffers (For the actual value for your 64-bit platform, see your machine notes. The maximum number of buffers on Solaris is 536,870,912.)

The **buffers** value specifies the maximum number of shared-memory buffers that the database server user threads have available for disk I/O on behalf of client applications. Therefore, the number of buffers that the database server requires depends on the applications. For example, if the database server accesses 15 percent of the application data 90 percent of the time, you need to allocate enough buffers to hold that 15 percent. Increasing the number of buffers can improve system performance.

Recommendation: Set the buffer space before you calculate other shared-memory parameters. On systems with a large amount of physical memory (4 GB or more), buffer space can be as much as 90 percent of physical memory.

Buffers and Read-Ahead

If you also want to perform read-ahead, increase the value of **buffers**. After you have configured all other shared-memory parameters, if you find that you can afford to increase the size of shared memory, increase the value of **buffers** until buffer space reaches the recommended 25 percent maximum.

Buffers and Smart Large Objects

If your databases contain smart large objects, you need to consider them when you calculate the value for **buffers**, because smart large objects are stored in the default page size buffer pool. If your applications frequently access smart large objects that are 2 kilobytes or 4 kilobytes in size, use the buffer pool to keep them in memory longer.

Use the following formula to increase the value of **buffers**:

```
Additional_BUFFERS = numcur_open_lo *
                     (lo_userdata / pagesize)
```

numcur_open_lo
: is the number of concurrently opened smart large objects that you can obtain from the **onstat -g smb fdd** option.

lo_userdata
: is the number of bytes of smart-large-object data that you want to buffer.

pagesize
: is the page size in bytes for the database server.

As a general rule, try to have enough buffers to hold two smart-large-object pages for each concurrently open smart large object. (The additional page is available for read-ahead purposes).

If the system uses lightweight I/O (as set by the access-mode constant LO_NOBUFFER), the system allocates the buffers from shared memory and does not store the smart large objects in the buffer pool. For information on access-mode flags and constants, see the chapter on “Working with Smart Large Objects of the Universal Data Option” in the *IBM Informix ESQL/C Programmer’s Manual*.

The lru_min_dirty Field

onconfig.std *value*	`lru_min_dirty=50`
syntax	`lru_min_dirty=`*percent_min*
units	Percent
range of values	`0` through `100` (fractional values are allowed)

The **lru_min_dirty** field specifies the percentage of modified pages in the LRU queues at which page cleaning is no longer mandatory. Page cleaners might continue cleaning beyond this point under some circumstances. If a field is specified out of the range of values, then the default of 50.00 percent is set.

The lru_max_dirty Field

onconfig.std *value*	`lru_max_dirty=60`
syntax	`lru_max_dirty=`*percent_max*
units	Percent
range of values	`0` through `100` (fractional values are allowed)

The **lru_max_dirty** field specifies the percentage of modified pages in the LRU queues at which the queue is cleaned. If a field is specified out of the range of values, then the default of 60.00 percent is set.

The size Field

onconfig.std *value*	`size=2K`
syntax	`size=`*size*
units	Kilobytes
range of values	`2` through `16`

The **size** field specifies the page size for the particular BUFFERPOOL line. The `K` is optional.

System Page Size

The system page size is the default page size and is platform-dependent on Dynamic Server.

You can use the following utilities to display the system page size.

Utility	**Description**
onstat -b	Displays the system page size, given as buffer size on the last line of the output
oncheck -pr	Checks the root-dbspace reserved pages and displays the system page size in the first section of its output
ON–Monitor (UNIX)	Displays the system page size under the the **Parameters > Initialize** option. Displays system

page size under the **Parameters > Shared-Memory** option, which does not require the database server to be running.

CKPTINTVL

onconfig.std *value*	`300`
units	Seconds
range of values	Any value greater than or equal to 0
takes effect	When the database server is shut down and restarted
refer to	The following material: • Checkpoints, in the shared-memory and fast-recovery chapters of the *IBM Informix Administrator's Guide* • Your *IBM Informix Performance Guide*

CKPTINTVL specifies the frequency, expressed in seconds, at which the database server checks to determine whether a checkpoint is needed. When a full checkpoint occurs, all pages in the shared-memory buffer pool are written to disk. When a fuzzy checkpoint occurs, nonfuzzy pages are written to disk, and the page numbers of fuzzy pages are recorded in the logical log.

If you set CKPTINTVL to an interval that is too short, the system spends too much time performing checkpoints, and the performance of other work suffers. If you set CKPTINTVL to an interval that is too long, fast recovery might take too long.

In practice, 30 seconds is the smallest interval that the database server checks. If you specify a checkpoint interval of `0`, the database server does not check if the checkpoint interval has elapsed. However, the database server still performs checkpoints. Other conditions, such as the physical log becoming 75 percent full, also cause the database server to perform checkpoints.

CLEANERS

onconfig.std *value*	1
units	Number of page-cleaner threads
range of values	1 through 128
takes effect	When the database server is shut down and restarted
utilities	**onstat -F** (See page 14-15.)
refer to	How the database server flushes data to disk, in the shared-memory chapter of the *IBM Informix Administrator's Guide*

CLEANERS specifies the number of page-cleaner threads available during the database server operation. By default, the database server always runs one page-cleaner thread. A general guideline is one page cleaner per disk drive. The value specified has no effect on the size of shared memory.

CONSOLE

onconfig.std *value*	On UNIX: **/dev/console** On Windows: **console.log**
range of values	*Pathname*
takes effect	When the database server is shut down and restarted
refer to	The system console in the chapter on database server administration in the *IBM Informix Administrator's Guide*

CONSOLE specifies the pathname and the filename for console messages.

DATASKIP

syntax	DATASKIP *state* [*dbspace1 dbspace2 ...*] The *state* entry is required. If *state* is `ON`, at least one *dbspace* entry is required.
onconfig.std *value*	None
if not present	`OFF`
separators	Space
range of values	`ALL` = Skip all unavailable fragments. `OFF` = Turn off DATASKIP. `ON` = Skip some unavailable fragments.
utilities	**onspaces -f** (See page 13-25.) **onstat -f** (See page 14-15.)
refer to	The following material: • "Specify DATASKIP Parameter" on page 13-25 • Your *IBM Informix Performance Guide*

DATASKIP lets you avoid points of media failure. This capability can result in higher availability for your data. To instruct the database server to skip some or all unavailable fragments, set this parameter. Whenever the database server skips over a dbspace during query processing, a warning is returned.

ESQL/C

The previously reserved SQLCA warning flag **sqlwarn.sqlwarn7** is set to `W` for IBM Informix ESQL/C

End of ESQL/C

Use the following syntax in the parameter line:

```
DATASKIP OFF
DATASKIP ON dbspace1 dbspace2...
DATASKIP ALL
```

Use the **-f** option of the **onspaces** utility to alter the value of the DATASKIP parameter at runtime.

An application can use the SQL statement SET DATASKIP to override the DATASKIP value that the ONCONFIG parameter or **onspaces** sets. If the application then executes the SQL statement SET DATASKIP DEFAULT, the DATASKIP value for that session returns to whatever value is currently set for the database server.

DBCREATE_PERMISSION

syntax	DBCREATE_PERMISSION *value*
onconfig.std *value*	Not set
units	a username
takes effect	When the database server is shut down and restarted

DBCREATE_PERMISSION restricts the permission to create databases to the specified user. You can include multiple copies of DBCREATE_PERMISSION in the ONCONFIG file to give additional users permission to create databases.

The informix user always has permission to create databases. To restrict the ability to create databases to the informix user, add the following line to the ONCONFIG file:

```
DBCREATE_PERMISSION informix
```

DB_LIBRARY_PATH

syntax	DB_LIBRARY_PATH *value*
onconfig.std *value*	None
if not present	The database server can load external modules from any location
range of values	List of path names (up to 512 bytes)
separators	Comma
takes effect	When the database server is shut down and restarted

The DB_LIBRARY_PATH configuration parameter specifies a comma-separated list of valid directory prefix locations from which the database server can load external modules, such as DataBlade Modules. You can also include server environment variables, such as $INFORMIXDIR.

You must specify paths for external modules exactly as they are registered with Dynamic Server. Relative paths or paths that include double periods (..) are not valid. External modules in the file systems that are not specified by this parameter cannot be loaded. This list is scanned prior to loading C language modules.

If you set this parameter, you must also include the string `$INFORMIXDIR/extend` as part of the value. If the string `$INFORMIXDIR/extend` is not included in DB_LIBRARY_PATH, IBM-supplied DataBlade Modules, the BladeManager, Large Object Locator DataBlade module functions, and DataBlade modules that you created with the DataBlade Developer's Kit will not load.

DBSERVERALIASES

onconfig.std *value*	None
if not present	None
separators	Comma
range of values	Up to 128 lowercase characters for each dbserver alias. Up to 32 values separated by commas. The value for DBSERVERALIASES follows the same rules as the DBSERVERNAME parameter (see "DBSERVERNAME" on page 1-22).
takes effect	When the database server is shut down and restarted. In addition, you might need to update the **sqlhosts** file or registry of each database server.
MaxConnect users	To use MaxConnect with more than one communication protocol, specify additional dbservernames in the DBSERVERALIASES parameter in the ONCONFIG file. The value of the **INFORMIXSERVER** environment variable on the client must match either the DBSERVERNAME or one of the entries of the DBSERVERALIASES parameter.
refer to	The following topics in the chapter on client/server communications in the *IBM Informix Administrator's Guide*: • ONCONFIG parameters for connectivity • Using multiple connection types

DBSERVERALIASES specifies a list of alternative dbservernames. If the database server supports more than one communication protocol (for example, both an IPC mechanism and the TCP network protocol), you must describe each valid connection to the database server with an entry in the **sqlhosts** file or registry. DBSERVERALIASES lets you assign multiple aliases to a database server, so each entry in the **sqlhosts** file or registry can have a unique name.

Important:: You can specify up to 32 DBSERVERALIASES for a database server. If you attempt to define more than 32 DBSERVERALIASES, a warning message displays twice on the console. If you attempt to specify the DBSERVERALIASES all on one line, and the line exceeds 512 bytes, the excess bytes are truncated.

For each alternate name listed in DBSERVERALIASES, the database server starts an additional listener thread. If you have many client applications connecting to the database server, you can distribute the connection requests between several listener threads and reduce connection time. To take advantage of the alternate connections, instruct some of your client applications to use a CONNECT TO *dbserveralias* statement instead of CONNECT TO *dbservername.*

DBSERVERNAME

onconfig.std *value*	None
if not present	On UNIX: *hostname*

	On Windows: ol_*hostname* (The *hostname* variable is the name of the host computer.)
range of values	Up to 128 lowercase characters DBSERVERNAME must begin with a letter and can include any printable character except the following characters: • Uppercase characters • A field delimiter (space or tab) • A newline character • A comment character • A hyphen, minus, or @ character
takes effect	When the database server is shut down and restarted. The **sqlhosts** file or registry of each database server that communicates with this database server might need to be updated. In addition, the **INFORMIXSERVER** environment variable for all users might need to be changed.
MaxConnect users	The value of the **INFORMIXSERVER** environment variable on the client must match either the DBSERVERNAME or one of the entries of the DBSERVERALIASES parameter.
refer to	DBSERVERNAME configuration parameter in the chapter on client/server communications in the *IBM Informix Administrator's Guide*

When you install the database server, specify the dbservername. DBSERVERNAME specifies a unique name associated with this specific occurrence of the database server. The value of DBSERVERNAME is called the *dbservername*. Each dbservername is associated with a communication protocol in the **sqlhosts** file or registry. If the database server uses multiple communication protocols, additional values for dbservername must be defined with the DBSERVERALIASES configuration parameter.

Client applications use dbservername in the **INFORMIXSERVER** environment variable and in SQL statements such as CONNECT and DATABASE, which establish a connection to a database server.

Important: To avoid conflict with other instances of Informix database servers on the same computer or node, it is recommended that you use DBSERVERNAME to assign a dbservername explicitly.

DBSPACETEMP

onconfig.std *value*	None
if not present	ROOTNAME
separators	Comma or colon (no white space)
range of values	The list of dbspaces can contain standard dbspaces, temporary dbspaces, or both. Use a colon or comma to separate the dbspaces in your list. The length of the list cannot exceed 254 characters.

takes effect	When the database server is shut down and restarted
environment variable **DBSPACETEMP**	Specifies dbspaces that the database server uses to store temporary tables for a particular session. If **DBSPACETEMP** is not set, the default location is the root dbspace.
utilities	**onspaces -t** (See page 13-6.) **onstat -d flags** field (See page 14-11.)
refer to	The following material: • What is a temporary table, in the chapter on data storage in the *IBM Informix Administrator's Guide* • *IBM Informix Guide to SQL: Reference* • The order of precedence that the database server uses when it creates implicit sort files, in the *IBM Informix Performance Guide* • The order of precedence of the default locations where the database server stores logged and unlogged temporary tables in the *IBM Informix Guide to SQL: Reference*.

DBSPACETEMP specifies a list of dbspaces that the database server uses to globally manage the storage of temporary tables. DBSPACETEMP improves performance by enabling the database server to spread out I/O for temporary tables efficiently across multiple disks. The database server also uses temporary dbspaces during backups to store the before-images of data that are overwritten while the backup is occurring.

DBSPACETEMP can contain dbspaces with a non-default page size, but all of the dbspaces in the DBSPACETEMP list must have the same page size. For more information about dbspaces in non-default buffer pools, see "BUFFERPOOL" on page 1-14.

If a client application needs to specify an alternative list of dbspaces to use for its temporary-table locations, the client can use the **DBSPACETEMP** environment variable to list them.

Important: The dbspaces that you list in the DBSPACETEMP configuration parameter must consist of chunks that are allocated as raw UNIX devices. On Windows, you can create temporary dbspaces in *NTFS* files.

If both standard and temporary dbspaces are listed in the DBSPACETEMP configuration parameter or environment variable, the following rules apply:

- Sort, backup, implicit, and nonlogging explicit temporary tables are created in temporary dbspaces if adequate space exists.
- Explicit temporary tables created without the WITH NO LOG option are created in standard (rather than temporary) dbspaces.

When you create a temporary dbspace with ISA or with the **onspaces** utility, the database server does not use the newly created temporary dbspace until you perform the following steps.

To enable the database server to use the new temporary dbspace:

1. Add the name of a new temporary dbspace to your list of temporary dbspaces in the DBSPACETEMP configuration parameter, the **DBSPACETEMP** environment variable, or both.
2. Restart the database server with the **oninit** command (UNIX) or restart the database server service (Windows).

If you use the **DBSPACETEMP** environment variable to create a temporary dbspace in a user session, the change takes effect immediately and overrides the DBSPACETEMP value in the **ONCONFIG** file.

Using Hash Join Overflow and DBSPACETEMP

Dynamic Server uses an operating-system directory or file to direct any overflow that results from the following database operations if you do not set the **DBSPACETEMP** environment variable or DBSPACETEMP configuration parameter. You can specify the operating-system directory or file in the following ways:

- SELECT statement with GROUP BY clause
- SELECT statement with ORDER BY clause
- Hash-join operation
- Nested-loop join operation
- Index builds

If you do not set the **DBSPACETEMP** environment variable or DBSPACETEMP configuration parameter, the database server directs any overflow that results from the preceding operations to the operating-system directory or file that you specify in one of the following variables:

UNIX Only

- On UNIX, the operating-system directory or directories that the **PSORT_DBTEMP** environment variable specifies, if it is set. If **PSORT_DBTEMP** is not set, the database server writes sort files to the operating-system file space in the **tmp** directory.

End of UNIX Only

Windows Only

- On Windows, the directory specified in TEMP or TMP in the User Environment Variables window in **Control Panel > System**.

End of Windows Only

DD_HASHMAX

onconfig.std *value*	None
units	Maximum number of tables in a hash bucket
range of values	Positive integers
takes effect	When the database server is shut down and restarted
utilities	Use a text editor to modify the configuration file.

refer to	The following material: • Configuration effects on memory, in your *IBM Informix Performance Guide* • "DD_HASHSIZE" on page 1-26

DD_HASHMAX specifies the maximum number of tables in each hash bucket in the data-dictionary cache. A *hash bucket* is the unit of storage (typically a page) whose address is computed by the hash function. A hash bucket contains several records.

For example, if DD_HASHMAX is 10 and DD_HASHSIZE is 100, you can store information about 1000 tables in the data-dictionary cache, and each hash bucket can have a maximum of 10 tables.

DD_HASHSIZE

onconfig.std *value*	None
units	Number of hash buckets or lists
range of values	Any positive prime number
takes effect	When the database server is shut down and restarted
utilities	Use a text editor to modify the configuration file.
refer to	The following material: • Configuration effects on memory, in your *IBM Informix Performance Guide* • "DD_HASHMAX" on page 1-25

DD_HASHSIZE specifies the number of hash buckets or lists in the data-dictionary cache.

DEADLOCK_TIMEOUT

onconfig.std *value*	`60`
units	Seconds
range of values	Positive integers
takes effect	When the database server is shut down and restarted
utilities	**onstat -p** *dltouts* field (See 14-86.)
refer to	Configuration parameters used in two-phase commits, in the chapter on multiphase commit protocols in the *IBM Informix Administrator's Guide*

DEADLOCK_TIMEOUT specifies the maximum number of seconds that a database server thread can wait to acquire a lock. Use this parameter only for distributed queries that involve a remote database server. Do not use this parameter for nondistributed queries.

DEF_TABLE_LOCKMODE

onconfig.std *value*	PAGE
if not present	PAGE
range of values	PAGE = sets lock mode to page for new tables ROW = sets lock mode to row for new tables
takes effect	When the database server is shut down and restarted
environment variable	**IFX_DEF_TABLE_LOCKMODE**
refer to	The following material: • Environment variables in the *IBM Informix Guide to SQL: Reference* • Setting lock modes, in the *IBM Informix Guide to SQL: Tutorial* • Configuring lock mode, in the *IBM Informix Performance Guide*

If DEF_TABLE_LOCKMODE = ROW, it sets the lock mode to row for every newly created table for all sessions that are connected to logging or nonlogging databases. This parameter has no effect on the lock mode for existing tables.

The rules of precedence for setting the lock mode are as follows.

Precedence	Command
1 (highest)	CREATE TABLE or ALTER TABLE statement that use the LOCK MODE clause
2	**IFX_DEF_TABLE_LOCKMODE** environment variable set on the client side
3	**IFX_DEF_TABLE_LOCKMODE** environment variable set on the server side
4	DEF_TABLE_LOCKMODE value in ONCONFIG file
5 (lowest)	Default behavior (page-level locking)

DIRECTIVES

onconfig.std *value*	1
range of values	0 or 1
takes effect	When the database server is shut down and restarted
environment variable	**IFX_DIRECTIVES**
refer to	The following material: • Environment variables in the *IBM Informix Guide to SQL: Reference* • SQL directives, in the *IBM Informix Guide to SQL: Syntax* • Performance impact of directives, in your *IBM Informix Performance Guide*

The DIRECTIVES parameter enables or disables the use of SQL directives. SQL directives allow you to specify behavior for the query optimizer in developing query plans for SELECT, UPDATE, and DELETE statements.

Set DIRECTIVES to `1`, which is the default value, to enable the database server to process directives. Set DIRECTIVES to `0` to disable the database server from processing directives. Client programs also can set the **IFX_DIRECTIVES** environment variable to `ON` or `OFF` to enable or disable processing of directives by the database server. The setting of the **IFX_DIRECTIVES** environment variable overrides the setting of the DIRECTIVES configuration parameter. If you do not set the **IFX_DIRECTIVES** environment variable, all sessions for a client inherit the database server configuration for processing SQL directives.

DISABLE_B162428_XA_FIX

onconfig.std *value*	None
units	Integer
range of values	`0` = (Default) Frees transactions only when an XA rollback is called 1 = Frees transactions if transaction rollback for other than an XA rollback
takes effect	When the database server is shut down and restarted
refer to	*IBM Informix Guide to SQL: Reference*

Set DISABLE_B162428_XA_FIX to 1 to immediately free all global transactions after a transaction rollback, which is the default for Dynamic Server 9.40 and earlier versions. The default behavior for Dynamic Server 10.0 is to free global transactions after an xa_rollback is called, and this behavior is required to confirm to the XA state table that a transaction can be freed only after xa_rollback is called. Setting DISABLE_B162428_XA_FIX to 1 ensures that applications written for the earlier version of Dynamic server work properly.

You can override the DISABLE_B162428_XA_FIX configuration parameter for a client session with the **IFX_XASTDCOMPLIANCE_XAEND** environment variable. Setting **IFX_XASTDCOMPLIANCE_XAEND** to 1 will free transactions only when an XA rollback is called. Setting **IFX_XASTDCOMPLIANCE_XAEND** to 0 will free transactions if the transaction rollback is for other than an XA rollback.

DRIDXAUTO

onconfig.std *value*	0
range of values	`0`= Off 1 = On
utilities	**onstat** (See "The onstat -g dri Option" on page 14-28.)
takes effect	When the database server is shut down and restarted

Specifies whether the primary High-Availability Data Replication (HDR) server automatically starts index replication if the secondary HDR server detects a corrupted index. To enable automatic index replication, set the value of the

DRIDXAUTO configuration parameter to 1. You can alter the value of DRIDXAUTO for a running server instance without restarting the instance using the **onmode -d idxauto** command. However, the **onmode -d idxauto** command will not change the value of the DRIDXAUTO parameter in the ONCONFIG file. For more information, see "Replicate an Index with Data-Replication" on page 10-10.

DRAUTO

onconfig.std *value*	`0`
range of values	`0` signifies OFF = Do not automatically switch the server type in the HDR environment. `1` signifies `RETAIN_TYPE` = Automatically switch secondary to standard during an HDR failure. Switch back to secondary when restarting HDR. `2` signifies `REVERSE_TYPE` = Automatically switch secondary to standard on an HDR failure. Switch to primary (and switch original primary to secondary) when restarting HDR.
takes effect	When shared memory is initialized
utilities	ON-Monitor > Parameters > data-Replication > Auto **onstat** (See "The onstat -g dri Option" on page 14-28.)

DRAUTO determines how a secondary database server reacts to an HDR failure. This parameter should have the same value on both HDR servers.

If DRAUTO is set to `OFF`, the secondary database server remains a secondary database server in read-only mode when an HDR failure occurs.

If DRAUTO is set to either `RETAIN_TYPE` or `REVERSE_TYPE`, the secondary database server switches to type standard automatically when an HDR failure is detected. If DRAUTO is set to `RETAIN_TYPE`, the original secondary database server switches back to type secondary when the HDR connection is restored. If DRAUTO is set to `REVERSE_TYPE`, the original secondary database server switches to type primary when the HDR connection is restored, and the original primary switches to type secondary.

Use this parameter carefully. A network failure (that is, when the primary database server does not really fail, but the secondary database server perceives network slowness as an HDR failure) can cause the two database servers to become out of synch.

DRINTERVAL

onconfig.std *value*	`30`
units	Seconds
range of values	`-1`, `0`, and positive integer values
takes effect	When the database server is shut down and restarted

utilities	**onstat** (See "The onstat -g dri Option" on page 14-28.)
refer to	When log records are sent, in the chapter on High-Availability Data Replication in the *IBM Informix Administrator's Guide*

DRINTERVAL specifies the maximum interval in seconds between flushing of the high-availability data-replication buffer. To update synchronously, set the parameter to `-1`.

DRLOSTFOUND

onconfig.std *value*	On UNIX: **/usr/etc/dr.lostfound** On Windows: *drive*:**\informix\etc\dr.lostfound**
range of values	*Pathname*
takes effect	When the database server is shut down and restarted
utilities	**onstat** (See "The onstat -g dri Option" on page 14-28.)
refer to	Lost-and-found transactions, in the chapter on High-Availability Data Replication in the *IBM Informix Administrator's Guide*

DRLOSTFOUND specifies the pathname to the **dr.lostfound.timestamp** file. This file contains transactions committed on the primary database server but not committed on the secondary database server when the primary database server experiences a failure. The file is created with a time stamp appended to the filename so that the database server does not overwrite another lost-and-found file if one already exists.

This parameter is not applicable if updating between the primary and secondary database servers occurs synchronously (that is, if DRINTERVAL is set to `-1`).

DRTIMEOUT

onconfig.std *value*	`30`
units	Seconds
range of values	Positive integers
takes effect	When the database server is shut down and restarted
utilities	**onstat** (See "The onstat -g dri Option" on page 14-28.)
refer to	How High-Availability Data Replication failures are detected, in the chapter on High-Availability Data Replication in the *IBM Informix Administrator's Guide*

DRTIMEOUT applies only to high-availability data-replication pairs. This value specifies the length of time, in seconds, that a database server in a high-availability data-replication pair waits for a transfer acknowledgment from the other database server in the pair. Use the following formula to calculate DRTIMEOUT:

```
DRTIMEOUT = wait_time / 4
```

In this formula, *wait_time* is the length of time, in seconds, that a database server in a high-availability data-replication pair must wait before it assumes that a high-availability data-replication failure occurred.

For example, suppose you determine that *wait_time* for your system is 160 seconds. Use the preceding formula to set DRTIMEOUT as follows:

```
DRTIMEOUT = 160 seconds / 4 = 40 seconds
```

DS_HASHSIZE

onconfig.std *value*	None
if not present	31
units	Number of hash buckets or lists
range of values	Any positive prime number
takes effect	When the database server is shut down and restarted
refer to	The following material: • *IBM Informix Performance Guide* for how to monitor and tune the data-distribution cache • "DS_POOLSIZE" on page 1-33

The DS_HASHSIZE parameter specifies the number of hash buckets in the data-distribution cache that the database server uses to store and access column statistics that the UPDATE STATISTICS statement generates in the MEDIUM or HIGH mode.

Use DS_HASHSIZE and DS_POOLSIZE to improve performance of frequently executed queries in a multiuser environment.

For information on configuration parameters for UDR cache, see "PC_HASHSIZE" on page 1-62 and "PC_POOLSIZE" on page 1-62.

DS_MAX_QUERIES

onconfig.std *value*	On UNIX: None On Windows: 32
if not present	`num_cpu_vps * 2 * 128`
units	Number of queries
range of values	Minimum = 1 Maximum = 8,388,608 (8 megabytes)
utilities	**onmode -Q** (See page 10-15.) **onstat -g mgm** (See "The onstat -g mgm Option" on page 14-47.)
refer to	The following material: • "Specifying the Number of CPU VPs" on page 1-89 • Parallel database query in your *IBM Informix Performance Guide*

DS_MAX_QUERIES is the maximum number of PDQ queries that can run concurrently. The Memory Grant Manager (MGM) reserves memory for a query based on the following formula:

```
memory_reserved = DS_TOTAL_MEMORY *
                    (PDQ-priority / 100) *
                 (MAX_PDQPRIORITY / 100)
```

The value of PDQPRIORITY is specified in either the **PDQPRIORITY** environment variable or the SQL statement SET PDQPRIORITY.

DS_MAX_SCANS

onconfig.std *value*	1,048,576 or (1024 * 1024)
units	Number of PDQ scan threads
range of values	10 through (1024 * 1024)
utilities	**onmode -S** (See page 10-15.) **onstat -g mgm** (See "The onstat -g mgm Option" on page 14-47.)
refer to	Parallel database query in your *IBM Informix Performance Guide*

DS_MAX_SCANS limits the number of PDQ scan threads that the database server can execute concurrently. When a user issues a query, the database server apportions some number of scan threads, depending on the following values:

- The value of PDQ priority (set by the environment variable **PDQPRIORITY** or the SQL statement SET PDQPRIORITY)
- The ceiling that you set with DS_MAX_SCANS
- The factor that you set with MAX_PDQPRIORITY
- The number of fragments in the table to scan (*nfrags* in the formula)

The Memory Grant Manager (MGM) tries to reserve scan threads for a query according to the following formula:

```
reserved_threads = min (nfrags, (DS_MAX_SCANS *
                    PDQPRIORITY / 100 *
                  MAX_PDQPRIORITY / 100) )
```

If the DS_MAX_SCANS part of the formula is greater than or equal to the number of fragments in the table to scan, the query is held in the ready queue until as many scan threads are available as there are table fragments. Once underway, the query executes quickly because threads are scanning fragments in parallel.

For example, if *nfrags* equals 24, DS_MAX_SCANS equals 90, **PDQPRIORITY** equals 50, and MAX_PDQPRIORITY equals 60, the query does not begin execution until *nfrags* scan threads are available. Scanning takes place in parallel.

If the DS_MAX_SCANS formula falls below the number of fragments, the query might begin execution sooner, but the query takes longer to execute because some threads scan fragments serially.

If you reduce DS_MAX_SCANS to 40 in the previous example, the query needs fewer resources (12 scan threads) to begin execution, but each thread needs to scan two fragments serially. Execution takes longer.

DS_NONPDQ_QUERY_MEM

onconfig.std	128
units	Kilobytes
range of values	From 128 Kilobytes to 25 percent of the value of DS_TOTAL_MEMORY
takes effect	When the database server is initialized
utilities	**onstat -g mgm** (See "The onstat -g mgm Option" on page 14-47.) **onmode** ON-Monitor

Use the DS_NONPDQ_QUERY_MEM configuration parameter to increase the amount of memory that is a available for a query that is not a Parallel Database Query (PDQ). (You can only use this parameter if PDQ priority is set to zero.) If you specify a value for the DS_NONPDQ_QUERY_MEM parameter, determine and adjust the value based on the number and size of table rows.

The DS_NONPDQ_QUERY_MEM value is calculated during database server initialization based on the calculated DS_TOTAL_MEMORY value. If during the processing of the DS_NONPDQ_QUERY_MEM, the database server changes the value that you set, the server sends a message in this format:

```
DS_NONPDQ_QUERY_MEM recalculated and changed from old_value Kb to new_value Kb.
```

In the message, *old_value* represents the value that you assigned to DS_NONPDQ_QUERY_MEM in the user configuration file, and *new_value* represents the value determined by the database server.

The value for DS_NONPDQ_QUERY_MEM can be changed using the **onmode -wf** option or superseded for a session with the **onmode -wm** option. For more information about **onmode**, see "Dynamically Change Certain Connection, PDQ, and Memory Parameters" on page 10-19.

DS_POOLSIZE

onconfig.std *value*	None
if not present	127
units	Maximum number of entries in the data-distribution cache
range of values	Any positive value
takes effect	When the database server is shut down and restarted
refer to	The following material: • *IBM Informix Performance Guide* for how to monitor and tune the data-distribution cache • "DS_HASHSIZE" on page 1-31

The DS_POOLSIZE parameter specifies the maximum number of entries in each hash bucket in the data-distribution cache that the database server uses to store and access column statistics that the UPDATE STATISTICS statement generates in the MEDIUM or HIGH mode.

Use DS_HASHSIZE and DS_POOLSIZE to improve performance of frequently executed queries in a multi-user environment.

For information on configuration parameters for UDR cache, see "PC_HASHSIZE" on page 1-62 and "PC_POOLSIZE" on page 1-62.

DS_TOTAL_MEMORY

onconfig.std *value*	On UNIX: None On Windows: `4,096`
if not present	If SHMTOTAL=`0` and DS_MAX_QUERIES is set, `DS_TOTAL_MEMORY = DS_MAX_QUERIES * 128`. If SHMTOTAL=`0` and DS_MAX_QUERIES is not set, `DS_TOTAL_MEMORY = num_cpu_vps * 2 * 128`.
units	Kilobytes
range of values	If DS_MAX_QUERY is set, the minimum value is `DS_MAX_QUERY * 128`. If DS_MAX_QUERY is not set, the minimum value is `num_cpu_vps * 2 * 128`. Maximum value for 32-bit platform: 2 gigabytes Maximum value for 64-bit platform: 4 gigabytes
utilities	**onmode -M** (See page 10-15.) **onstat -g mgm** (See "The onstat -g mgm Option" on page 14-47.)
refer to	The following material: • Your *IBM Informix Performance Guide* for the algorithms • "SHMTOTAL" on page 1-72 • "SHMVIRTSIZE" on page 1-73 • "Specifying the Number of CPU VPs" on page 1-89 • The maximum memory available on your platform, in the machine notes

DS_TOTAL_MEMORY specifies the amount of memory available for PDQ queries. It should be smaller than the computer physical memory, minus fixed overhead such as operating-system size and buffer-pool size.

Do not confuse DS_TOTAL_MEMORY with the configuration parameters SHMTOTAL and SHMVIRTSIZE. SHMTOTAL specifies all the memory for the database server (total of the resident, virtual, and message portions of memory). SHMVIRTSIZE specifies the size of the virtual portion. DS_TOTAL_MEMORY is part of SHMVIRTSIZE.

For OLTP applications, set DS_TOTAL_MEMORY to between 20 and 50 percent of the value of SHMTOTAL in kilobytes.

For applications that involve large decision-support (DSS) queries, increase the value of DS_TOTAL_MEMORY to between 50 and 80 percent of SHMTOTAL. If you use your database server for DSS queries exclusively, set this parameter to 90 and 100 percent of SHMTOTAL.

Set the DS_TOTAL_MEMORY configuration parameter to any value not greater than the quantity (`SHMVIRTSIZE - 10 megabytes`).

Algorithm for DS_TOTAL_MEMORY

The database server derives a value for DS_TOTAL_MEMORY when you do not set DS_TOTAL_MEMORY, or if you set it to an inappropriate value. For information on the algorithms, see configuration effects on memory utilization in your *IBM Informix Dynamic Server Performance Guide*.

DUMPCNT (UNIX)

onconfig.std *value*	1
if not present	1
units	Number of assertion failures
range of values	Positive integers
takes effect	When the database server is shut down and restarted
refer to	Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

DUMPCNT specifies the number of assertion failures for which one database server thread dumps shared memory or generates a core file by calling **gcore**. An assertion is a test of some condition or expression with the expectation that the outcome is true. For example, the following statement illustrates the concept of an assertion failure:

```
if (a != b)
    assert_fail("a != b");
```

DUMPCORE (UNIX)

onconfig.std *value*	0
range of values	0 = Do not dump core image. 1 = Dump core image.
takes effect	When the database server is shut down and restarted
refer to	Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

DUMPCORE controls whether assertion failures cause a virtual processor to dump a core image. The core file is left in the directory from which the database server was last invoked. (The DUMPDIR parameter has no impact on the location of the core file.)

Warning: When *DUMPCORE* is set to *1, an asse*rtion failure causes a virtual processor to dump a core image, which in turn causes the database server to abort. Set *DUMPCORE* only for debugging purposes in a controlled environment.

DUMPDIR

3	*onconfig.std value*	On UNIX: **/usr/informix/tmp**
3		On Windows: **%INFORMIXDIR%\tmp**
3	*if not present*	**$INFORMIXDIR/tmp**
	range of values	Any directory to which user **informix** has write access
	takes effect	When the database server is shut down and restarted
	refer to	Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

DUMPDIR specifies a directory in which the database server dumps shared memory, **gcore** files, or messages from a failed assertion. Because shared memory can be large, set DUMPDIR to a file system with a significant amount of space.

DUMPGCORE (UNIX)

onconfig.std *value*	0
range of values	0 = Do not dump **gcore**. 1 = Dump **gcore**.
takes effect	When the database server is shut down and restarted
refer to	Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

DUMPGCORE is used with operating systems that support **gcore**. If you set DUMPGCORE, but your operating system does not support **gcore**, messages in the database server message log indicate that an attempt was made to dump a core image, but the database server cannot find the expected file. (If your operating system does not support **gcore**, set DUMPCORE instead.)

If DUMPGCORE is set, the database server calls **gcore** whenever a virtual processor encounters an assertion failure. The **gcore** utility directs the virtual processor to dump a core image to the **core.pid.cnt** file in the directory that DUMPDIR specifies and continue processing.

The **pid** value is the process identification number of the virtual processor. The **cnt** value is incremented each time that this process encounters an assertion failure. The **cnt** value can range from 1 to the value of DUMPCNT. After that, no more core files are created. If the virtual processor continues to encounter assertion failures, errors are reported to the message log (and perhaps to the application), but no further diagnostic information is saved.

DUMPSHMEM (UNIX)

onconfig.std *value*	1
range of values	0 = Do not dump shared memory. 1 = Dump shared memory.
takes effect	When the database server is shut down and restarted
refer to	Collecting diagnostic information in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

DUMPSHMEM indicates that shared memory should be dumped on an assertion failure. All the shared memory that the database server uses is dumped; it is probably quite large. The shared-memory dump is placed in the **shmem.pid.cnt** file in the directory that DUMPDIR specifies.

The **pid** value is the process identification number for the virtual processor. The **cnt** value is incremented each time that this virtual processor encounters an assertion failure. The **cnt** value can range from 1 to the value of DUMPCNT. After the value of DUMPCNT is reached, no more files are created. If the database server continues to detect inconsistencies, errors are reported to the message log (and perhaps to the application), but no further diagnostic information is saved.

DYNAMIC_LOGS

onconfig.std *value*	None (this parameter is not in the **onconfig.std** file)
if not present	2 (Default)
range of values	0 = Turn off dynamic-log allocation. 1 = Set off the "log file required" alarm and pause to allow manual addition of a logical-log file. You can add a log file immediately after the current log file or to the end of the log file list. 2 = Turn on dynamic-log allocation. When the database server dynamically adds a log file, it sets off the "dynamically added log file" alarm.
takes effect	When the database server is shut down and restarted
utilities	"Add a Logical-Log File" on page 12-2
refer to	The following material: • "LTXEHWM" on page 1-48 • "LTXHWM" on page 1-49 • Logical logs in the *IBM Informix Administrator's Guide*

If DYNAMIC_LOGS is 2, the database server automatically allocates a new log file when the next active log file contains an open transaction. Dynamic-log allocation prevents long transaction rollbacks from hanging the system.

If you want to choose the size and location of the new logical-log file, set DYNAMIC_LOGS to 1. Use the **onparams -a** command with the size (**-s**), location (**-d dbspace**), and **-i** options to add a log file after the current log file.

Even when DYNAMIC_LOGS is turned off, you do not have the same risks as in previous database server versions. In Version 9.3 and later, if the database server hangs from a long transaction rollback, you can shut down the database server, set DYNAMIC_LOGS to 1 or 2, and then restart the database server.

Important: If you are using *Enterprise Replication* with dynamic log allocation, set *LTXEHWM* to no higher than *70*.

Enterprise Replication Configuration Parameters

The following configuration parameters apply to Enterprise Replication. For more information, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

Configuration Parameter	Description
CDR_DBSPACE	Specifies the dbspace where the **syscdr** database is created.
CDR_DSLOCKWAIT	Specifies the number of seconds that the Datasync (data synchronization) component waits for database locks to be released.
CDR_ENV	Sets the Enterprise Replication environment variables **CDR_LOGDELTA, CDR_PERFLOG**, **CRD_ROUTER,** or **CDR_RMSCALEFACT.**
CDR_EVALTHREADS	Specifies the number of grouper evaluator threads to create when Enterprise Replication starts and enables parallelism.
CDR_MAX_DYNAMIC_LOGS	Specifies the number of dynamic log file requests that Enterprise Replication can make in one server session.
CDR_NIFCOMPRESS	Specifies the level of compression that the database server uses before sending data from the source database server to the target database server.
CDR_QDATA_SBSPACE	Specifies the list of up to 32 names of sbspaces that Enterprise Replication uses to store spooled transaction row data.
CDR_QHDR_DBSPACE	Specifies the location of the dbspace that Enterprise Replication uses to store the transaction record headers spooled from the send and receive queues.
CDR_QUEUEMEM	Specifies the maximum amount of memory that is used for the send and receive queues.
CDR_SERIAL	Controls generating values for SERIAL and SERIAL8 columns in tables defined for replication. Use this parameter to generate SERIAL column primary keys.

CDR_SUPPRESS_ATSRISWARN
: Specifies the Datasync error and warning code numbers to be suppressed in the ATS and RIS files.

ENCRYPT_CDR
: Specifies the level of Enterprise Replication encryption.

ENCRYPT_CIPHERS
: Specifies the ciphers to use for Enterprise Replication encryption.

ENCRYPT_MAC
: Specifies the level of message authentication coding to use with Enterprise Replication encryption.

ENCRYPT_MACFILE
: Specifies the message authentication coding key files to use with Enterprise Replication encryption.

ENCRYPT_SWITCH
: Defines the frequency at which ciphers and secret keys are re-negotiated for Enterprise Replication encryption.

EXT_DIRECTIVES

onconfig.std *value*
: `0`

range of values
: `0, 1, 2`

takes effect
: When the database server is shut down and restarted

environment variable
: **IFX_EXTDIRECTIVES**

refer to
: The following material:
- Environment variables and information about the **sysdirectives** system catalog table, in the *IBM Informix Guide to SQL: Reference*
- SQL directives, in the *IBM Informix Guide to SQL: Syntax*
- Using external optimizer directives, in the *IBM Informix Performance Guide*

The EXT_DIRECTIVES configuration parameter enables or disables the use of external SQL directives. Enable external directives by using the EXT_DIRECTIVES configuration parameter in combination with the client-side **IFX_EXTDIRECTIVES** environment variable as follows:

Value	Explanation
0 (default)	Off. The directive cannot be enabled even if **IFX_EXTDIRECTIVES** is on.
1	On. The directive can be enabled for a session if **IFX_EXTDIRECTIVES** is on.
2	On. The directive can be used even if **IFX_EXTDIRECTIVES** is not set.

The setting of the **IFX_EXTDIRECTIVES** environment variable overrides the setting of the EXT_DIRECTIVES configuration parameter. If you do not set the **IFX_EXTDIRECTIVES** environment variable, all sessions for a client inherit the database server configuration for processing external directives.

EXTSHMADD

onconfig.std *value*	8192
range of values	1024 through 524,288
units	Kilobytes
takes effect	When the database server is shut down and restarted
utilities	onstat -g seg

EXTSHMADD specifies the size of extension virtual segments that you add. Other virtual segment additions are based on the size that is specified in the SHMADD configuration parameter.

FAST_RESTART_CKPT_FUZZYLOG

onconfig.std *value*	The FAST_RESTART_CKPT_FUZZYLOG parameter does not need to be in the **onconfig.std** file.
range of values	0 (default) = Disable the flushing of dirty fuzzy pages to the physical log at checkpoint. 1 = Enable the flushing of dirty fuzzy pages to the physical log at checkpoint.
takes effect	At the checkpoint that occurs after the parameter is enabled. If the total number of unflushed, dirty fuzzy pages exceeds 20 percent of the total physical log space, the pages will not be written to the physical log. If server fails after this checkpoint, crash recovery receives no performance benefit.
refer to	Information on fast recovery and alternative fast restart recovery options for fuzzy operations in the *IBM Informix Administrator's Guide*.

The FAST_RESTART_CKPT_FUZZYLOG parameter and the FAST_RESTART_PHYSLOG parameter enable the database server to perform physical logging on fuzzy checkpoints during the roll-forward (log replay) phase of recovery, thus decreasing recovery time. You can use either parameter or both when using fuzzy checkpoints.

The database server must be online when you enable the FAST_RESTART_CKPT_FUZZYLOG parameter.

FAST_RESTART_PHYSLOG

onconfig.std *value*	The FAST_RESTART_PHYSLOG parameter does not need to be in the **onconfig.std** file.
range of values	0 (default) = Disable physical logging on fuzzy checkpoints during the roll-forward (log replay) phase of recovery. 1 = Enable physical logging on fuzzy checkpoints during the roll-forward (log replay) phase of recovery, thus decreasing recovery time.

takes effect	Immediately. If the total number of unflushed, fuzzy dirty pages exceeds 20 percent of the total physical log space, the pages will not be written to the physical log. However, if the database server fails before the next checkpoint performs, maximum fast-recovery performance does not occur because the database server did not log all of the fuzzy updates in the checkpoint intervals.
refer to	Information on fast recovery and alternative fast restart recovery options for fuzzy operations in the *IBM Informix Administrator's Guide*

The FAST_RESTART_PHYSLOG parameter and the FAST_RESTART_CKPT_FUZZYLOG parameter enable the database server to perform physical logging on fuzzy checkpoints during the roll-forward (log replay) phase of recovery, thus decreasing recovery time. You can use either parameter or both when using fuzzy checkpoints.

Only use the FAST_RESTART_PHYSLOG parameter if the buffer pool is at least 25 percent larger than the physical buffer size. The buffer pool must be large enough to hold the physical log, log pages, and other pages read during recovery. If the buffer pool is not configured correctly, fast recovery performance is compromised.

The extra physical logging that occurs when the database server uses the FAST_RESTART_PHYSLOG parameter affects runtime performance. If you do not want to sacrifice runtime performance or if you do not want to increase the buffer size, use the FAST_RESTART_CKPT_FUZZYLOG parameter to reduce some recovery time.

After enabling the FAST_RESTART_PHYSLOG parameter by setting it to 1 , you can initiate fast recovery using the **oninit** utility. Simply execute **oninit** without any options.

The database server must be online when you enable the FAST_RESTART_PHYSLOG parameter.

FILLFACTOR

onconfig.std *value*	90
units	Percent
range of values	1 through 100
takes effect	When the index is built. Existing indexes are not changed. To use the new value, the indexes must be rebuilt.
refer to	"Structure of B-Tree Index Pages" on page 3-16

FILLFACTOR specifies the degree of index-page fullness. A low value provides room for growth in the index. A high value compacts the index. If an index is full (100 percent), any new inserts result in splitting nodes. You can also set the FILLFACTOR as an option on the CREATE INDEX statement. The setting on the CREATE INDEX statement overrides the ONCONFIG file value.

HETERO_COMMIT

onconfig.std *value*	`0`
range of values	`1` = Enable heterogeneous commit. `0` = Disable heterogeneous commit.
takes effect	When the database server is shut down and restarted
refer to	The following material: • Heterogeneous commit protocol, in the chapter on multiphase commit protocols in the *IBM Informix Administrator's Guide* • *IBM Informix Enterprise Gateway Manager User Manual*

The HETERO_COMMIT configuration parameter specifies whether or not the database server is prepared to participate with IBM Informix Gateway products in heterogeneous commit transactions. Setting HETERO_COMMIT to 1 allows a single transaction to update *one* non-Informix database (accessed with any of the Gateway products) and one or more Informix databases.

If HETERO_COMMIT is `0`, a single transaction can update databases as follows:
- One or more Informix databases and no non-Informix databases
- One non-Informix database and no Informix databases

You can read data from any number of Informix and non-Informix databases, regardless of the setting of HETERO_COMMIT.

IFX_EXTEND_ROLE

4 **onconfig.std** *value*	`1`
range of values	`1` or `On` = Enables the EXTEND role so that administrators can grant privileges to a user to create or drop a UDR that has the EXTERNAL clause. `0` or `Off` (default) = Disables the EXTEND role so that any user can register an external routine.
refer to	Information on security for external routines in the *IBM Informix Administrator's Guide*

Your database system administrator (DBSA), by default user **informix**, uses the IFX_EXTEND_ROLE parameter to implement security measures that establish which users can register DataBlade user-defined routines (UDRs). This prevents unauthorized users from registering the external routines.

ISM_DATA_POOL and ISM_LOG_POOL

The ISM_DATA_POOL and ISM_LOG_POOL parameters control where IBM Informix Storage Manager stores backed-up data and logical logs. For information on these parameters, see the *IBM Informix Backup and Restore Guide* or the *IBM Informix Storage Manager Administrator's Guide*.

Java Configuration Parameters

The following configuration parameters allow you to use J/Foundation, which incorporates an embedded Java virtual machine on the database server. For more information on these parameters, see *J/Foundation Developer's Guide*.

Configuration Parameter	Description
AFCRASH	When the 0x10 bit is on for AFCRASH, all the messages that the Java Virtual Machine generates are logged into the **JVM_vpid** file, where *vpid* is the process ID of the Java virtual processor. This file is stored in the directory where the JVPLOG file is stored.
JDKVERSION	Version number of the Java Development Kit (JDK) or Java Runtime Environment (JRE) release
JVPDEBUG	When set to 1, writes tracing messages to the JVPLOG file
JVPHOME	Directory where the classes of the IBM Informix JDBC Driver are installed
JVPLOGFILE	Absolute pathname for your Java VP log files
JVPPROPFILE	Absolute pathname for the Java VP properties file
JVPJAVAVM	Libraries to use for the Java Virtual Machine (JVM)
JVPJAVAHOME	Directory where the Java Runtime Environment (JRE) for the database server is installed
JVMTHREAD	Thread package (green or native) to use for the JVM
JVPJAVALIB	Path from **JVPJAVAHOME** to the location of the Java VM libraries
JVPCLASSPATH	Initial Java class path setting
VPCLASS JVP	Number of Java virtual processors that the database server should start. (See "VPCLASS" on page 1-85.)

LISTEN_TIMEOUT

onconfig.std *value*	10
Units	Seconds
takes effect	When the database server is stopped and restarted
utilities	**onmode -wf** **onmode-wm**
refer to	The Security chapter in the *IBM Informix Administrator's Guide*

LISTEN_TIMEOUT specifies the number of seconds the server waits for a connection. It can be set to a lower number to guard against faulty connection requests that might indicate a Denial of Service attack. See also information about the MAX_INCOMPLETE_CONNECTIONS configuration parameter on page 1-50.

Depending on the machine capability of holding the threads (in number), you can configure MAX_INCOMPLETE_CONNECTIONS to a higher value and depending on the network traffic, you can set LISTEN_TIMEOUT to a lower value to reduce the chance that an attack can reach the maximum limit.

Both the LISTEN_TIMEOUT and the MAX_INCOMPLETE_CONNECTIONS configuration parameters can be changed using the **onmode -wf** option or superseded for a session with the **onmode -wm** option. For more information about **onmode**, see "Dynamically Change Certain Connection, PDQ, and Memory Parameters" on page 10-19.

LOCKS

onconfig.std *value*	`2,000`
units	Number of locks in the internal lock table
range of values	2,000 through 8,000,000
takes effect	When the database server is shut down and restarted
utilities	**onstat -k** (See page 14-80.)
refer to	The following material: • The memory and locking chapters in your *IBM Informix Performance Guide* • The shared memory chapter in the *IBM Informix Administrator's Guide*

LOCKS specifies the initial size of the lock table. The lock table holds an entry for each lock that a session uses. If the number of locks that sessions allocate exceeds the value of LOCKS, the database server increases the size of the lock table.

Although each additional lock takes up just 44 bytes of resident shared memory, locks can become a resource drain if you have a limited amount of shared memory. For example, if you set LOCKS to `1,000,000`, the database server allocates 40 megabytes of resident shared memory for locks.

Tip: When you drop a database, a lock is acquired and held on each table in the database until the database is dropped. For more information on the DROP DATABASE statement, see the "*IBM Informix Guide to SQL: Syntax*."

LOGBUFF

onconfig.std *value*	`32`
units	Kilobytes
range of values	32 kilobytes through (32767 * `page size` / 1024) kilobytes
takes effect	When the database server is shut down and restarted
utilities	**onstat -l buffer** field, second section (See page 14-82.)
refer to	Logical-log buffer, in the shared-memory chapter of the *IBM Informix Administrator's Guide*

LOGBUFF specifies the size in kilobytes for the three logical-log buffers in shared memory. Triple buffering permits user threads to write to the active buffer while one of the other buffers is being flushed to disk. If flushing is not complete by the time the active buffer fills, the user thread begins writing to the third buffer.

It is recommended that you set LOGBUFF to 16 or 32 kilobytes, or perhaps 64 kilobytes for heavy workloads. Choose a value for LOGBUFF that is evenly divisible by the page size. If the value of LOGBUFF is not evenly divisible by the page size, the database server rounds down the size to the nearest value that is evenly divisible by the page size.

If you log user data in smart large objects, increase the size of the log buffer to make the system more efficient. The database server logs only the portion of a smart-large-object page that changed.

Important: The database server uses the *LOGBUFF* parameter to set the size of internal buffers that are used during recovery. If you set *LOGBUFF* too high, the database server can run out of memory and shut down during recovery.

To set the system page size, use one of the utilities listed in "System Page Size" on page 1-18.

LOGFILES

onconfig.std *value*	6
if not present	6
units	Number of logical-log files
range of values	3 through 32,767 (integers only)
takes effect	During disk initialization and when you add a new log file. You add a new log with one of the following utilities.
utilities	**onparams** (See page 12-1.)
refer to	The following topics in the *IBM Informix Administrator's Guide*: • Size of logical-log files, in the chapter on the logical log • Adding or dropping a logical-log file, in the chapter on managing the logical log

LOGFILES specifies the number of logical-log files that the database server creates during disk initialization. To change the number of logical-log files, add or drop logical-log files.

If you use ISA or **onparams** to add or drop log files, the database server automatically updates LOGFILES.

LOGSIZE

onconfig.std *value*	2000
if not present	UNIX: 1500 Windows: 500

units	Kilobytes
range of values	Minimum = `200` Maximum =`(ROOTSIZE - PHYSFILE - 512 - (63 * ((pagesize)/1024))) / LOGFILES` The *pagesize* value is platform dependent.
takes effect	When the database server is shut down and restarted. The size of log files added after shared memory is initialized reflects the new value, but the size of existing log files does not change.
utilities	**onparams** See "Change Physical-Log Parameters" on page 12-3.
refer to	The following topics in the *IBM Informix Administrator's Guide*: • Size of the logical log and logging smart large objects, in the chapter on the logical log • Changes to LOGSIZE or LOGFILES, in the chapter on managing logical logs • "LTXHWM" on page 1-49

LOGSIZE specifies the size that is used when logical-log files are created. It does not change the size of existing logical-log files. The total logical-log size is `LOGSIZE * LOGFILES`.

To verify the page size that the database server uses on your platform, use one of the utilities listed in "System Page Size" on page 1-18.

LOGSIZE for Smart Large Objects

If you declare logging for a smart-large-object column, you must ensure that the logical log is considerably larger than the amount of data logged during inserts or updates.

Important: The database server cannot back up open transactions. If many transactions are active, the total logging activity should not force open transactions to the log backup files. For example, if your log size is 1000 kilobytes and the high-watermark is 60 percent, do not use more than 600 kilobytes of the logical log for the smart-large-object updates. The database server starts rolling back the transaction when it reaches the high-watermark of 600 kilobytes.

LTAPEBLK

onconfig.std *value*	`32`
units	Kilobytes
range of values	Values greater than (*page size*/`1024`) To obtain the page size, see the commands listed in "System Page Size" on page 1-18
takes effect	For **ontape**: When you execute **ontape** For **onload** and **onunload**: When the database server is shut down and restarted

refer to	The following material: • Using **ontape**, in the *IBM Informix Backup and Restore Guide* • Using onload and onunload, in the *IBM Informix Migration Guide* • "TAPEBLK" on page 1-81

LTAPEBLK specifies the block size of the device to which the logical logs are backed up when you use **ontape** for dbspace backups. LTAPEBLK also specifies the block size for the device to which data is loaded or unloaded when you use the -l option of **onload** or **onunload**. If you are using **onload** or **onunload**, you can specify a different block size at the command line.

Specify LTAPEBLK as the largest block size permitted by your tape device. The database server does not check the tape device when you specify the block size. Verify that the LTAPEDEV tape device can read the block size that you specify. If not, you might not be able to read from the tape.

UNIX Only

The UNIX **dd** utility can verify that the LTAPEDEV tape device can read the block size. It is available with most UNIX systems.

End of UNIX Only

LTAPEDEV

onconfig.std *value*	On UNIX: **/dev/tapedev** On Windows: **\\.\TAPE1**
if not present	On UNIX: **/dev/null** On Windows: **nul**
takes effect	For **ontape**: when the database server is shut down and restarted, if set to **/dev/null** on UNIX or **nul** on Windows. When you execute **ontape**, if set to a tape device. For **onload** and **onunload**: when the database server is shut down and restarted
refer to	The following material: • How to set and change the LTAPEDEV value for **ontape** and how LTAPEDEV affects ON–Bar, in the *IBM Informix Backup and Restore Guide* • Using **onload** or **onunload**, in the *IBM Informix Migration Guide* • "TAPEDEV" on page 1-81

LTAPEDEV specifies the device to which the logical logs are backed up when you use **ontape** for backups. LTAPEDEV also specifies the device to which data is loaded or unloaded when you use the **-l** option of **onload** or **onunload**. If you are using LTAPEDEV to specify a device for **onunload** or **onload**, the same information for TAPEDEV is relevant for LTAPEDEV.

Warning: Do not set LTAPEDEV to **/dev/null** or **nul** when you use ON–Bar to back up logical logs.

LTAPESIZE

onconfig.std *value*	10,240
units	Kilobytes
range of values	Positive integers
takes effect	For **ontape**: when you execute **ontape** For **onload** and **onunload**: when the database server is shut down and restarted
refer to	The following material: • Using **ontape**, in the *IBM Informix Backup and Restore Guide* • Using **onload** or **onunload**, in the *IBM Informix Migration Guide* • "TAPESIZE" on page 1-82

LTAPESIZE specifies the maximum tape size of the device to which the logical logs are backed up when you use **ontape** for backups. LTAPESIZE also specifies the maximum tape size of the device to which data is loaded or unloaded when you use the **-l** option of **onload** or **onunload**. If you are using **onload** or **onunload**, you can specify a different tape size on the command line. If you want to use the full capacity of a tape, set LTAPESIZE to 0.

LTXEHWM

onconfig.std *value*	None (not present in **onconfig.std**)
if not present	90 (if DYNAMIC_LOGS is set to 1 or 2) 60 (if DYNAMIC_LOGS is set to 0)
units	Percent
range of values	LTXHWM through 100
takes effect	When the database server is shut down and restarted
refer to	The following material: • "DYNAMIC_LOGS" on page 1-37 • "LTXHWM" on page 1-49 • Setting high-watermarks for rolling back long transactions, in the chapter on managing logical logs in the *IBM Informix Administrator's Guide*

A *transaction* is *long* if it is not committed or rolled back when it reaches the long-transaction high-watermark. LTXEHWM specifies the *long-transaction, exclusive-access, high-watermark*. When the logical-log space reaches the LTXEHWM threshold, the long transaction currently being rolled back is given *exclusive* access to the logical log.

If your system runs out of log space before the rollback completes, lower the LTXEHWM value.

If you do not want too many logical logs to be added, LTXEHWM should be set to a smaller value (around 60). If dynamic logging is turned off (DYNAMIC_LOGS = 0), LTXEHWM should be set lower (around 50) to avoid running out of logical space.

Tip: To allow users to continue to access the logical logs, even during a long transaction rollback, set *LTXEHWM* to *100*. Set *DYNAMIC_LOGS* to *1* or *2* so that the database server can add log files as needed to complete the transaction or rollback.

LTXHWM

onconfig.std *value*	None (not present in **onconfig.std**)
if not present	`80` (if DYNAMIC_LOGS is set to `1` or `2`) `50` (if DYNAMIC_LOGS is set to `0`)
units	Percent
range of values	1 through 100
takes effect	When the database server is shut down and restarted
refer to	The following material: • "DYNAMIC_LOGS" on page 1-37 • "LTXEHWM" on page 1-48 • Setting high-watermarks for rolling back long transactions, in the chapter on managing logical logs in the *IBM Informix Administrator's Guide*

LTXHWM specifies the long-transaction high-watermark. The *long-transaction high-watermark* is the percentage of available log space that, when filled, triggers the database server to check for a long transaction. When the logical-log space reaches the LTXHWM threshold, the database server starts rolling back the transaction. If you decrease the LTXHWM value, increase the size or number of log files to make rollbacks less likely.

If DYNAMIC_LOGS is set to `1` or `2`, the database server adds as many logs are needed to complete the rollback.

If you do not want too many logical logs to be added, LTXHWM should be set to a smaller value (around 60). If dynamic logging is turned off (DYNAMIC_LOGS = 0), LTXHWM should be set lower (around 50) to avoid running out of logical space.

Warning: If you set both LTXHWM and LTXEHWM to 100, long transactions are never aborted. Although you can use this configuration to your advantage, you should set LTXHWM to below 100 for normal database server operations.

If you set LTXHWM to `100`, the database server issues a warning message:

```
LTXHWM is set to 100%. This long transaction high water mark will never be reached.
Transactions will not be aborted automatically by the server, regardless of their length.
```

If the transaction hangs, follow the instructions for recovering from a long transaction hang, in the chapter on managing logical-log files in the *IBM Informix Administrator's Guide*.

MAX_INCOMPLETE_CONNECTIONS

onconfig.std *value*	1024
units	Number of listener threads
takes effect	When the database server is stopped and restarted
utilities	**onmode -wf** **onmode-wm**
refer to	The Security chapter in the *IBM Informix Administrator's Guide*

Use MAX_INCOMPLETE_CONNECTIONS to specify the maximum number of incomplete connections in a session. After this number is reached, an error message is written in the online message log stating that the server might be under a Denial of Service attack. See also information about the LISTEN_TIMEOUT configuration parameter on page 1-43.

Depending on the machine capability of holding the threads (in number), you can configure MAX_INCOMPLETE_CONNECTIONS to a higher value and depending on the network traffic, you can set LISTEN_TIMEOUT to a lower value to reduce the chance that an attack can reach the maximum limit.

Both the MAX_INCOMPLETE_CONNECTIONS and the LISTEN_TIMEOUT configuration parameters can be changed using the **onmode -wf** option or superseded for a session with the **onmode -wm** option. For more information about **onmode**, see "Dynamically Change Certain Connection, PDQ, and Memory Parameters" on page 10-19.

MAX_PDQPRIORITY

onconfig.std *value*	`100`
if not present	`100`
range of values	0 through 100
takes effect	On all user sessions
utilities	**onmode -D** **onstat -g mgm** (See "The onstat -g mgm Option" on page 14-47.)
refer to	The following material: • The chapter on using PDQ, in the *IBM Informix Performance Guide* • "Change Decision-Support Parameters" on page 10-15

MAX_PDQPRIORITY limits the PDQ resources that the database server can allocate to any one DSS query. MAX_PDQPRIORITY is a factor that is used to scale the value of PDQ priority set by users. For example, suppose that the database administrator sets MAX_PDQPRIORITY to `80`. If a user sets the **PDQPRIORITY** environment variable to `50` and then issues a query, the database server silently processes the query with a PDQ priority of `40`.

You can use the **onmode** utility to change the value of MAX_PDQPRIORITY while the database server is online.

In Dynamic Server, PDQ resources include memory, CPU, disk I/O, and scan threads. MAX_PDQPRIORITY lets the database administrator run decision support concurrently with OLTP, without a deterioration of OLTP performance. However, if MAX_PDQPRIORITY is too low, the performance of decision- support queries can degrade.

You can set MAX_PDQPRIORITY to one of the following values.

Value	Database Server Action
0	Turns off PDQ. DSS queries use no parallelism.
1	Fetches data from fragmented tables in parallel (parallel scans) but uses no other form of parallelism.
100	Uses all available resources for processing queries in parallel.
number	An integer between 0 and 100. Sets the percentage of the user-requested PDQ resources actually allocated to the query.

MaxConnect Configuration Parameters

Before you start IBM Informix MaxConnect, you need to specify the following configuration parameters in the **IMCCONFIG** file. This file contains both start-time and runtime parameters.

Configuration Parameter	Description
IMCLOG	Specifies the pathname of the MaxConnect log file.
IMCTRANSPORTS	Specifies the number of TCP network connections (transports) between MaxConnect and the database server.
IMCWORKERDELAY	Determines the time that worker threads wait to accumulate packets before they perform an aggregated send.
IMCWORKERTHREADS	Specifies the number of worker threads for MaxConnect.

MaxConnect uses the following environment variables. For more information, see the section on the configuration file in the *IBM Informix MaxConnect User's Guide*:

- **INFORMIXDIR**
- **INFORMIXSERVER**
- **INFORMIXSQLHOSTS**
- **IMCADMIN**
- **IMCCONFIG**
- **IMCSERVER**

MIRROR

onconfig.std *value*	0
range of values	0 = disable mirroring 1 = enable mirroring

takes effect	When the database server is shut down and restarted
utilities	**onstat -d** *flags* field (See page 14-11.)
refer to	The following topics in the *IBM Informix Administrator's Guide*: • Mirroring critical data in the chapter on where is data stored • Enabling mirroring in the chapter on using mirroring

The MIRROR parameter indicates whether mirroring is enabled for the database server. It is recommended that you mirror the root dbspaces and the critical data as part of initialization. Otherwise, leave mirroring disabled. If you later decide to add mirroring, you can edit your configuration file to change the parameter value.

You do not have to set the MIRROR configuration parameter to the same value on both database servers in the high-availability data-replication pair. You can enable or disable mirroring on either the primary or the secondary database server independently. Do not set the MIRROR configuration parameter to 1 unless you are using mirroring.

MIRROROFFSET

onconfig.std *value*	0
units	Kilobytes
range of values	Any value greater than or equal to 0
takes effect	When the database server is shut down and restarted
refer to	Mirroring the root dbspace during initialization, in the chapter on using mirroring in the *IBM Informix Administrator's Guide*

In Dynamic Server, MIRROROFFSET specifies the offset into the disk partition or into the device to reach the chunk that serves as the mirror for the initial chunk of the root dbspace.

MIRRORPATH

onconfig.std *value*	None
range of values	65 or fewer characters
takes effect	When the database server is shut down and restarted
refer to	The following material in the *IBM Informix Administrator's Guide*: • Mirroring the root dbspace during initialization, in the chapter on using mirroring • Using links, in the chapter on managing disk space

MIRRORPATH specifies the full pathname of the mirrored chunk for the initial chunk of the root dbspace. MIRRORPATH should be a link to the chunk pathname of the actual mirrored chunk for the same reasons that ROOTPATH is specified as a link. Similarly, select a short pathname for the mirrored chunk.

Setting Permissions (UNIX)

You must set the permissions of the file that MIRRORPATH specifies to `660`. The owner and group must both be **informix**.

If you use raw disk space for your mirror chunk on a UNIX platform, it is recommended that you define MIRRORPATH as a pathname that is a link to the initial chunk of the mirror dbspace, instead of entering the actual device name for the initial chunk.

MSGPATH

onconfig.std *value*	On UNIX: **/usr/informix/online.log** On Windows: **online.log**
if not present	On UNIX: **/dev/tty**
range of values	Pathname
takes effect	When the database server is shut down and restarted
utilities	**onstat -m** to view the message log (For more information, see page 14-84.)
refer to	Message log, in the chapter on overview of database server administration in the *IBM Informix Administrator's Guide*

MSGPATH specifies the full pathname of the message-log file. The database server writes status messages and diagnostic messages to this file during operation.

If the file that MSGPATH specifies does not exist, the database server creates the file in the specified directory. If the directory that MSGPATH specifies does not exist, the database server sends the messages to the system console.

If the file that MSGPATH specifies does exist, the database server opens it and appends messages to it as they occur.

MULTIPROCESSOR

onconfig.std *value*	`0`
if not present	Platform dependent
range of values	`0` = No multiprocessor `1` = Multiprocessor available
takes effect	When the database server is shut down and restarted
refer to	CPU virtual processors, in the chapter on virtual processors in the *IBM Informix Administrator's Guide*

If MULTIPROCESSOR is set to `0`, the parameters that set processor affinity are ignored. MULTIPROCESSOR specifies whether the database server performs locking in a manner that is suitable for a single-processor computer or a multiprocessor computer.

NETTYPE

syntax

`NETTYPE`
`protocol,poll_threads,connections,VP_class`
Specify the *protocol* as **iiippp** where:

iii=`[ipc|ipc|soc|tli]`
ppp=`[shm|str|tcp|spx|imc]`

The *protocol* value is required. You cannot use any white space in the fields, but you can omit trailing commas.

4 **onconfig.std** *values*

4 On UNIX: None
On Windows: `onsoctcp`, `drsoctcp`,`1`,`NET`

if not present

protocol:

4 On UNIX: **protocol** field from the **sqlhosts** file
4 (with or without the database server prefix of **on**,
4 **ol**, or **dr**)

On Windows: `onsoctcp`

number of poll_threads: `1` *number of connections*: `50`
VP_*class*: NET for DBSERVERALIASES;
CPU for DBSERVERNAME

separators

Commas

range of values

number of poll_threads:

On UNIX:
If VP_*class* is NET, a value greater than or equal to `1` If VP_*class* is CPU, `1` through `num_cpu_vps`

On Windows:
Any value greater than or equal to `1`

number of connections: `1` through `32,767` VP_*class*:
CPU = CPU VPs (on UNIX)
NET = Network VPs

takes effect

When the database server is shut down and restarted

utilities

onstat -g nsc (See page 14-16.)
onstat -g nss
onstat -g nta

refer to

The following sections in the *IBM Informix Administrator's Guide*:

- Network protocol entry, in the chapter on client/server communications
- Multiplexed connections, in the chapter on client/server communications
- Network virtual processors, in the chapter on virtual processors

- Should poll threads run on CPU or network virtual processors, in the chapter on virtual processors
- Monitoring and tuning the number of poll threads and connections, in the *IBM Informix Performance Guide*

Configuring MaxConnect in *IBM Informix MaxConnect User's Guide*

The NETTYPE parameter usually provides tuning options for the protocols that **dbservername** entries define in the **sqlhosts** file or registry.

Each **dbservername** entry in the **sqlhosts** file or registry is defined on either the DBSERVERNAME parameter or the DBSERVERALIASES parameter in the ONCONFIG file.

The NETTYPE configuration parameter describes a network connection as follows:
- The protocol (or type of connection)
- The number of poll threads assigned to manage the connection
- The expected number of concurrent connections
- The class of virtual processor that will run the poll threads

You can specify a NETTYPE parameter for each protocol that you want the database server to use. The following example illustrates NETTYPE parameters for two types of connections to the database server: a shared memory connection for local clients, and a network connection that uses sockets:

```
NETTYPE ipcshm,3,,CPU
NETTYPE soctcp,,20,NET
```

The NETTYPE parameter for the shared-memory connection (**ipcshm**) specifies three poll threads to run in CPU virtual processors. The number of connections is not specified, so it is set to `50`. The NETTYPE parameter for the sockets connection (**soctcp**) specifies that only 20 simultaneous connections are expected for this protocol and that one poll thread (because the number of poll threads is not specified) will run in a network virtual processor (in this case, NET).

Protocol

The protocol entry is the same as the **nettype** field in the **sqlhosts** file or registry, except that the database server prefix of **on** or **ol** is optional. The first three characters of the protocol entry specify the interface type, and the last three characters specify the IPC mechanism or the network protocol.

Number of Poll Threads

This field specifies the number of poll threads for a specific protocol. The default value of *poll_threads* is `1`.

If your database server has a large number of connections, you might be able to improve performance by increasing the number of poll threads. In general, each poll thread can handle approximately 200 to 250 connections.

Number of Connections

This field specifies the maximum number of connections per poll thread that can use this protocol at the same time. The default value of *connections* is `50`. If only a few connections will be using a protocol concurrently, you might save memory by explicitly setting the estimated number of connections.

Use this formula to calculate the maximum number of connections expected. For shared memory (**ipcshm**), double the number of connections.

```
connections = max_connections / poll threads
```

UNIX Only

The following example specifies 3 poll threads and 20 connections for a total of 60 shared-memory connections.

```
NETTYPE ipcshm,3,20,CPU
```

For all net types other than **ipcshm**, the poll threads dynamically reallocate resources to support more connections, as needed. Avoid setting the value for the number of concurrent connections to much higher than you expect. Otherwise, you might waste system resources.

End of UNIX Only

Windows Only

On Windows, the number of connections per poll thread is used for **ipcshm** connections. Other protocols ignore this value. Use NET virtual processors to run the poll threads.

End of Windows Only

Class of Virtual Processor

You can set the VP_*class* entry to specify either `CPU` or `NET`. However, the combined number of poll threads defined with the CPU VP class for all net types cannot exceed the maximum number of CPU VPS. You should carefully distinguish between poll threads for network connections and poll threads for shared memory connections, which should run one per CPU virtual processor. TCP connections should only be in network virtual processors, and you should only have the minimum needed to maintain responsiveness. Shared memory connections should only be in CPU virtual processors and should run in every CPU virtual processor.

Note: If you use the VP classes `tli`, `shm`, `str`, or `soc` in the settings for the VPCLASS configuration parameter, you must use the class of virtual processor class NET for the NETTYPE configurator parameter. For more information on the VPCLASS configuration parameter, see "VPCLASS Name" on page 1-87.

For more advice on whether to run the poll threads on CPU or NET virtual processors, refer to the chapter on virtual processors in the *IBM Informix Administrator's Guide*.

Default Values

It is recommended that you use NETTYPE to configure each of your connections. However, if you do not use NETTYPE, the database server uses the default values to create a single poll thread for the protocol. If the dbservername is defined by DBSERVERNAME, by default the poll thread is run by the CPU class. If the dbservername is defined by DBSERVERALIASES, the default VP class is NET.

Multiplexed Connections

To enable the database server to use multiplexed connections on UNIX, you must include a special NETTYPE parameter with the value `sqlmux`, as in the following example:

```
NETTYPE sqlmux
```

IBM Informix MaxConnect

If you are using IBM Informix MaxConnect, see the *IBM Informix MaxConnect User's Guide* for how to specify the fields in the NETTYPE parameter. The **ontliimc** and **onsocimc** protocols use TCP/IP to communicate with MaxConnect. You can use these protocols to either connect MaxConnect or the application clients to the database server.

OFF_RECVRY_THREADS

onconfig.std *value*	`10`
units	Number of recovery threads that run in parallel
range of values	Positive integers
takes effect	When the database server is shut down and restarted
refer to	The following material: • *IBM Informix Backup and Restore Guide* • *IBM Informix Performance Guide*

OFF_RECVRY_THREADS is the number of recovery threads used in logical recovery when the database server is offline (during a cold restore). This number of threads is also used to roll forward logical-log records in fast recovery.

Before you perform a cold restore, you can set the value of this parameter to approximately the number of tables that have a large number of transactions against them in the logical log. For single-processor computers or nodes, more than 30 to 40 threads is probably too many, because the overhead of thread management offsets the increase in parallel processing.

ON_RECVRY_THREADS

onconfig.std *value*	1
units	Number of recovery threads that run in parallel
range of values	Positive integers
takes effect	When the database server is shut down and restarted
refer to	The following material: • *IBM Informix Backup and Restore Guide* • *IBM Informix Performance Guide*

ON_RECVRY_THREADS is the maximum number of recovery threads that the database server uses for logical recovery when the database server is online (during a warm restore).

You can tune ON_RECVRY_THREADS to the number of tables that are likely to be recovered, because the logical-log records that are processed during recovery are assigned threads by table number. The maximum degree of parallel processing occurs when the number of recovery threads matches the number of tables being recovered.

With fuzzy checkpoints, fast recovery might take longer than with full checkpoints. To improve the performance of fast recovery, increase the number of fast-recovery threads with the ON_RECVRY_THREADS parameter.

ON-Bar Configuration Parameters

The following table lists the configuration parameters that apply exclusively to the ON–Bar backup and restore utility. For more information on these parameters, see the *IBM Informix Backup and Restore Guide*.

Configuration Parameter	Description
BAR_ACT_LOG	Specifies the location of the ON–Bar activity log file.
BAR_BSALIB_PATH	Specifies the pathname and filename of the XBSA shared library for the storage manager.
BAR_DEBUG	Specifies the level of debugging messages in the ON-Bar activity log.
BAR_DEBUG_LOG	Specifies the location of the ON-Bar debug log.
BAR_HISTORY	Specifies whether the **sysutils** database maintains a backup history.
BAR_MAX_BACKUP	Specifies the maximum number of backup processes per ON–Bar command.
BAR_NB_XPORT_COUNT	Specifies the number of shared-memory data buffers for each backup or restore process.
BAR_PROGRESS_FREQ	Specifies in minutes how frequently the backup or restore progress messages display in the activity log.
BAR_RETRY	Specifies how many times ON–Bar should retry a backup or restore operation.
BAR_XFER_BUF_SIZE	Specifies the size in pages of the buffers.
ISM_DATA_POOL	Specifies the volume pool that you use for backing up storage spaces.
ISM_LOG_POOL	Specifies the volume pool that you use for backing up logical logs.

ONDBSPACEDOWN

onconfig.std *value*	0
range of values	0, 1, 2
refer to	Monitoring the database server for disabling I/O errors, in the chapter on consistency checking in the *IBM Informix Administrator's Guide*

ONDBSPACEDOWN defines the action that the database server will take when any disabling event occurs on a noncritical dbspace. The following values are valid for this parameter.

Value	Description
0	Continue. Causes the database server to mark a noncritical dbspace down and continue whenever a disabling I/O error occurs on it.
1	Abort. Causes the database server to fail without allowing a checkpoint to occur whenever a disabling I/O error occurs on any dbspace. Critical dbspaces run only in this mode.
2	Wait. Causes the database server to hang all updating threads as soon as the next checkpoint request occurs after a disabling I/O occurs on a noncritical dbspace.

ONLIDX_MAXMEM

onconfig.std *value*	5120
units	Kilobytes
range of values	16 through 4294967295
takes effect	When the database server is shut down and restarted
utilities	**onmode -wf** **onmode-wm**

The ONLIDX_MAXMEM configuration parameter limits the amount of memory that is allocated to a single *preimage* pool and a single *updator* log pool. The preimage and updator log pools, **pimage**_*partnum* and **ulog**_*partnum*, are shared memory pools that are created when a CREATE INDEX ONLINE statement is executed. The pools are freed when the execution of the statement is completed.

If you specify a value for this parameter and then create a table, add rows to the table, and start to execute a CREATE INDEX ONLINE statement on a column, you can also perform other operations on the column, such as running UPDATE STATISTICS HIGH, without having memory problems.

The ONLIDX_MAXMEM configuration parameter can be changed using the **onmode -wf** option or superseded for a session with the **onmode -wm** option. For more information about **onmode**, see "Dynamically Change Certain Connection, PDQ, and Memory Parameters" on page 10-19.

OPCACHEMAX (UNIX)

onconfig.std *value*	`0`
if not present	`128`
units	Kilobytes
range of values	0 through `(4 * 1024 * 1024)`
takes effect	When the Optical Subsystem needs more memory
utilities	**onstat -O** (For more information, see page 14-85.)
refer to	The following material: • *IBM Informix Optical Subsystem Guide* • **INFORMIXOPCACHE** environment variable, in the *IBM Informix Guide to SQL: Reference*

OPCACHEMAX specifies the size of the memory cache for the Optical Subsystem. The database server stores pieces of TEXT or BYTE data in the memory cache before it delivers them to the subsystem. Use this parameter only if you use the Optical Subsystem.

The **INFORMIXOPCACHE** environment variable lets the client restrict the size of the optical cache that it uses.

OPTCOMPIND

onconfig.std *value*	`2`
range of values	`0` = When appropriate indexes exist for each ordered pair of tables, the optimizer chooses index scans (nested-loop joins), without consideration of the cost, over table scans (hash joins). This value ensures compatibility with previous versions of the database server. `1` = The optimizer uses costs to determine an execution path if the isolation level is not Repeatable Read. Otherwise, the optimizer chooses index scans (it behaves as it does for the value `0`). This setting is recommended for optimal performance. `2` = The optimizer uses cost to determine an execution path for any isolation level. Index scans are not given preference over table scans; the optimizer bases its decision purely on cost. This value is the default if the variable is not set.
refer to	The following material: • Your *IBM Informix Performance Guide* • **OPTCOMPIND** environment variable, in the *IBM Informix Guide to SQL: Reference* • SET ENVIRONMENT OPTCOMPIND statement to dynamically change the value of the OPTCOMPIND configuration parameter for a session, in the *IBM Informix Guide to SQL: Syntax*

OPTCOMPIND helps the optimizer choose an appropriate query plan for your application.

Tip: You can think of the name of the variable as arising from “OPTimizer COMPare (the cost of using) INDexes (with other methods).”

Because of the nature of *hash joins*, an application with isolation mode set to Repeatable Read might *temporarily* lock all records in tables that are involved in the join (even those records that fail to qualify the join) for each ordered set of tables. This situation leads to higher contention among connections. Conversely, nested-loop joins lock fewer records but provide inferior performance when the database server retrieves a large number of rows. Thus, both join methods offer advantages and disadvantages. A client application can also influence the optimizer in its choice of a join method.

OPT_GOAL

onconfig.std *value*	-1
range of values	0 or -1
takes effect	When the database server is shut down and restarted
refer to	The following manuals: • ALL_ROWS and FIRST_ROWS directives and on the SET OPTIMIZATION statement, in the *IBM Informix Guide to SQL: Reference* • **OPT_GOAL** environment variable, in the *IBM Informix Guide to SQL: Syntax* • Performance issues associated with setting an optimization goal, in the *IBM Informix Performance Guide*

The OPT_GOAL parameter enables you to specify one of the following optimization goals for queries:
- Optimize for FIRST ROWS
- Optimize for ALL ROWS

A value of `0` sets the optimization goal to FIRST_ROWS. A value of `-1` sets the optimization goal to ALL_ROWS, which is the default.

When you set the optimization goal to optimize for FIRST ROWS, you specify that you want the database server to optimize queries for perceived response time. In other words, users of interactive applications perceive response time as the time that it takes to display data on the screen. Setting the optimization goal to FIRST ROWS configures the database server to return the first rows of data that satisfy the query.

When you set the optimization goal to optimize for ALL ROWS, you specify that you want the database server to optimize for the total execution time of the query. Making ALL ROWS the optimization goal instructs the database server to process the total query as quickly as possible, regardless of how long it takes to return the first rows to the application.

You can specify the optimization goal in one of four ways:
- By query (SELECT statement)

  Use the ALL_ROWS and FIRST_ROWS directives.
- By session

  Use the SET OPTIMIZATION statement.
- By environment

  Set the **OPT_GOAL** environment variable.
- By database server

  Set the OPT_GOAL configuration parameter.

To determine the optimization goal, the database server examines the settings in the order shown. The first setting encountered determines the optimization goal.

For example, if a query includes the ALL_ROWS directive but the OPT_GOAL configuration parameter is set to FIRST_ROWS, the database server optimizes for ALL_ROWS, as the query specifies.

PC_HASHSIZE

onconfig.std *value*	None
range of values	Any positive non-prime number
takes effect	When the database server is shut down and restarted
refer to	Your *IBM Informix Performance Guide*

Use PC_HASHSIZE to specify the number of hash buckets in the caches that the database server uses.

PC_HASHSIZE applies to UDR cache only. For information on configuration parameters for other types of cache, see the "DS_POOLSIZE" on page 1-33 and the "DS_HASHSIZE" on page 1-31.

PC_POOLSIZE

onconfig.std *value*	None
range of values	Any positive value
takes effect	When the database server is shut down and restarted
refer to	Your *IBM Informix Performance Guide*

PC_POOLSIZE specifies the maximum number of UDRs stored in the UDR cache.

For information on configuration parameters for other types of cache, see the "DS_POOLSIZE" on page 1-33 and the "DS_HASHSIZE" on page 1-31.

PHYSBUFF

onconfig.std *value*	`32`
units	Kilobytes
range of values	4 kilobytes through (32767 * `page size / 1024`) kilobytes.
takes effect	When the database server is shut down and restarted
utilities	**onstat -l** *buffer* field, first section (For m ore information, see page 14-82.)
refer to	Physical-log buffer, in the shared-memory chapter of the *IBM Informix Administrator's Guide*

PHYSBUFF specifies the size in kilobytes of the two physical-log buffers in shared memory. Double buffering permits user threads to write to the active physical-log buffer while the other buffer is being flushed to the physical log on disk. The value of the PHYSBUFF parameter determines how frequently the database server needs to flush the physical-log buffer to the physical-log file. The recommended value for PHYSBUFF is 32 pages.

A write to the physical-log buffer is exactly one page in length. Choose a value for PHYSBUFF that is evenly divisible by the page size. If the value of PHYSBUFF is not evenly divisible by the page size, the database server rounds down the size to the nearest value that is evenly divisible by the page size.

The user-data portion of a smart large object does not pass through the physical-log buffers.

The system page size is platform-dependent on Dynamic Server. To obtain the system page size, use the commands listed in the table in “System Page Size” on page 1-18.

PHYSDBS

onconfig.std *value*	**rootdbs**
if not present	The dbspace that ROOTNAME specifies
units	A dbspace
range of values	Up to 128 characters. PHYSDBS must be unique, begin with a letter or underscore, and contain only letters, numbers, underscores, or **$** characters.
takes effect	When the database server is shut down and restarted
refer to	The following material: • “Change Physical-Log Parameters” on page 12-3 • Where the physical log is located, in the chapter on what is the physical log in the *IBM Informix Administrator's Guide* • Changing the physical-log location and size, in the chapter on managing the physical log in the *IBM Informix Administrator's Guide*

PHYSDBS specifies the name of the dbspace that contains the physical log. To reduce disk contention, you can move the physical log to a dbspace other than the root dbspace.

When you initialize disk space (**oninit -i**), the PHYSDBS value must be equal to the ROOTDBS value.

PHYSFILE

onconfig.std *value*	`2000`
if not present	`200`
units	Kilobytes
range of values	200 or more
takes effect	When the database server is shut down and restarted

refer to	The following topics in the *IBM Informix Administrator's Guide*: • Sizing the physical log, in the chapter on the physical log • Changing the physical-log location and size, in the chapter on managing the physical log

PHYSFILE specifies the size of the physical log.

PLOG_OVERFLOW_PATH

onconfig.std *value*	`None`
if not present	`$INFORMIXDIR/tmp`
takes effect	When the database server is brought up (shared memory is initialized)
refer to	Your *IBM Informix Administrator's Guide*

The PLOG_OVERFLOW_PATH parameter specifies the location of the file that is used during fast recovery if the physical log file overflows. The file is **plog_extend**.*servernum* and by default located in $INFORMIXDIR/tmp. Use the full pathname to specify a different location for the file with the PLOG_OVERFLOW_PATH parameter.

RA_PAGES

onconfig.std *value*	None
if not present	`4` if MULTIPROCESSOR is `0`; `8` if MULTIPROCESSOR is `1`
units	Number of data pages
range of values	RA_THRESHOLD through BUFFERS
takes effect	When the database server is shut down and restarted
refer to	The following material: • Configuring the database server to read ahead, in the shared-memory chapter of the *IBM Informix Administrator's Guide* • Calculating RA_PAGES and RA_THRESHOLD, in your *IBM Informix Performance Guide*

RA_PAGES specifies the number of disk pages to attempt to read ahead during sequential scans of data records. Read-ahead can greatly speed up database processing by compensating for the slowness of I/O processing relative to the speed of CPU processing.

This parameter works with the RA_THRESHOLD parameter. Specifying values that are too large can result in excessive buffer-caching activity.

RA_THRESHOLD

onconfig.std *value*	None

if not present	`RA_PAGES/2`
units	Number of data pages
range of values	0 through (`RA_PAGES - 1`)
takes effect	When the database server is shut down and restarted
refer to	The following material: • Configuring the database server to read ahead, in the shared-memory chapter of the *IBM Informix Administrator's Guide* • Calculating RA_PAGES and RA_THRESHOLD, in your *IBM Informix Performance Guide*

RA_THRESHOLD is used with RA_PAGES when the database server reads during sequential scans of data records. RA_THRESHOLD specifies the read-ahead threshold; that is, the number of unprocessed data pages in memory that signals the database server to perform the next read-ahead.

If the value of RA_THRESHOLD is greater than the value of RA_PAGES, RA_THRESHOLD has a value of `RA_PAGES/2`.

Specifying values that are too large for RA_PAGES and RA_THRESHOLD can result in excessive buffer-caching activity.

RESIDENT

onconfig.std *value*	`0`
range of values	`-1 to 99` `0` = off `1` = lock the resident segment only `n` = lock the resident segment and the next *n*-1 virtual segments `-1` = lock all resident and virtual segments `99` = lock the resident segment and the next 98 virtual segments Certain platforms have different values. For information, see your machine notes.
if not present	`0`
takes effect	When the database server is shut down and restarted
utilities	**onmode -r** (see "Change Shared-Memory Residency" on page 10-8)
refer to	The following topics in the *IBM Informix Administrator's Guide* for a discussion of residency: • Resident portion of shared memory, in the shared-memory chapter • Setting database server shared-memory configuration parameters, in the chapter on managing shared memory

The RESIDENT parameter specifies whether resident and virtual segments of shared memory remain resident in operating-system physical memory.

Some systems allow you to specify that the resident portion of shared memory must stay (be resident) in memory at all times. If your operating system supports forced residency, you can specify that resident and virtual segments of shared memory not be swapped to disk.

Warning: Before you decide to enforce residency, verify that the amount of physical memory available is sufficient to execute all required operating-system and application processes. If insufficient memory is available, a system hang could result that requires a reboot.

RESTARTABLE_RESTORE

onconfig.std *value*	`ON`
if not present	`ON`
range of values	`OFF` = restartable restore is disabled `ON` = restartable restore is enabled
takes effect	When the database server is shut down and restarted
refer to	*IBM Informix Backup and Restore Guide*

If you set RESTARTABLE_RESTORE to `ON`, you enable the database server to restart a failed physical or cold logical restore at the point at which the failure occurred. To perform a restartable restore with ON–Bar, use the **onbar -RESTART** command.

Increase the size of your physical log if you plan to use restartable restore. For more information, see “PHYSFILE” on page 1-63. Although a restartable restore slows down the logical restore if many logs need to be restored, you save a lot of time from not having to repeat the entire restore.

Important: If the database server fails during a warm logical restore, you must repeat the entire restore. If the database server is still running, use **onbar -r -l** to complete the restore.

If you do a cold restore on systems that are not identical, you can assign new pathnames to chunks, and you can rename devices for critical chunks during the restore. You must perform a level-0 archive after the rename and restore operation completes. For details, see the *IBM Informix Backup and Restore Guide*

The database server uses physical recovery and logical recovery to restore data as follows:

- **Physical recovery.** The database server writes data pages from the backup media to disk. This action leaves the storage spaces consistent to the point at which it was originally backed up. However, the backup times for each storage space are usually different. A restartable restore is restartable to the level of a storage space. If only some chunks of a storage space are restored when the restore fails, the entire storage space needs to be recovered again when you restart the restore.

- **Logical recovery.** The database server replays logical-log records on media to bring all the storage spaces up to date. At the end of logical recovery, all storage spaces are consistent to the same point.

ROOTNAME

onconfig.std *value*	**rootdbs**
units	A dbspace
range of values	Up to 128 characters. ROOTNAME must begin with a letter or underscore and must contain only letters, numbers, underscores, or **$** characters.
takes effect	When disk is initialized (destroys all data)
refer to	Allocating disk space, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*

ROOTNAME specifies a name for the root dbspace for this database server configuration.

The name must be unique among all dbspaces that the database server manages. It is recommended that you select a name that is easily recognizable as the root dbspace.

ROOTOFFSET

onconfig.std ***value***	`0`
units	Kilobytes
range of values	Any value greater than or equal to 0
takes effect	When disk is initialized (destroys all data)
refer to	Allocating raw disk space on UNIX, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*

ROOTOFFSET specifies the offset into an allocation of disk space (file, disk partition, or device) at which the initial chunk of the root dbspace begins.

UNIX Only

On some UNIX platforms, it is not valid to set ROOTOFFSET to `0`. When this parameter is set incorrectly, you must reinitialize disk space and reload data to resume proper operation of the database server. Before you configure the database server, always check your machine notes file for information about proper settings.

End of UNIX Only

ROOTPATH

onconfig.std *value*	On UNIX: **/dev/online_root** On Windows: None
range of values	*Pathname*
takes effect	When disk is initialized (destroys all data)

refer to	The following material in the chapter on managing disk space in the *IBM Informix Administrator's Guide* • Allocating disk space • Creating links for raw devices

ROOTPATH specifies the full pathname, including the device or filename, of the initial chunk of the root dbspace. ROOTPATH is stored in the reserved pages as a chunk name.

On UNIX, you must set the permissions of the file that ROOTPATH specifies to `660`, and the owner and group must both be **informix**. On Windows, a member of the **Informix-Admin** group must own the file that ROOTPATH specifies.

UNIX Only

If you use unbuffered disk space for your initial chunk on UNIX, it is recommended that you define ROOTPATH as a pathname that is a link to the initial chunk of the root dbspace instead of entering the actual device name for the initial chunk.

End of UNIX Only

ROOTSIZE

onconfig.std *value*	UNIX: `30,000` Windows: `50,000`
if not present	`0`
units	Kilobytes
range of values	`0` through maximum capacity of the storage device
takes effect	When disk is initialized (destroys all data)
refer to	Calculating the size of the root dbspace, in the chapter on where is data stored in the *IBM Informix Administrator's Guide*

ROOTSIZE specifies the size of the initial chunk of the root dbspace, expressed in kilobytes. The size that you select depends on your immediate plans for your database server.

To change ROOTSIZE after you initialize the database server, completely unload and reload your data.

SBSPACENAME

onconfig.std *value*	None
if not present	`0`
range of values	Up to 128 characters. SBSPACENAME must be unique, begin with a letter or underscore, and contain only letters, digits, underscores, or **$** characters.
takes effect	When shared memory is reinitialized
utilities	**onspaces -c -S**

refer to	The following material: • Using **onspaces** to "Create an Sbspace or Temporary Sbspace" on page 13-9 • "SBSPACETEMP" on page 1-69 • "SYSSBSPACENAME" on page 1-80 • "Sbspace Structure" on page 3-23 • What is an sbspace, in the chapter on data storage in the *IBM Informix Administrator's Guide* • Altering sbspace characteristics, in the chapter on managing data on disk in the *IBM Informix Administrator's Guide* • Assigning a smart large object to an sbspace, in the section on the CREATE TABLE and ALTER TABLE statements, in the *IBM Informix Guide to SQL: Syntax* • Creating an sbspace for Enterprise Replication usage, in the *IBM Informix Dynamic Server Enterprise Replication Guide* • Using multirepresentational data, in the *IBM Informix DataBlade API Programmer's Guide*

SBSPACENAME specifies the name of the default sbspace. If your database tables include smart-large-object columns that do not explicitly specify a storage space, that data is stored in the sbspace that SBSPACENAME specifies.

You must create the default sbspace with the **onspaces -c -S** utility before you can use it. The database server validates the name of the default sbspace when one of the following occurs:

- You specify the default sbspace as the storage option for a CLOB or BLOB column in the PUT clause of the CREATE TABLE or ALTER TABLE statement.
- The database server attempts to write a smart large object to the default sbspace when no sbspace was specified for the column.
- You store multirepresentational data in the default sbspace.

JAVA Language Support

If you are using IBM Informix Dynamic Server with J/Foundation, you must provide a smart large object where the database server can store the Java archive (JAR) files. These JAR files contain your Java user-defined routines (UDRs). It is suggested that when you use Java UDRs, you create separate sbspaces for storing smart large objects.

End of JAVA Language Support

Warning: When you use Enterprise Replication, you must set the *CDR_QDATA_SBSPACE* parameter and create the sbspace before you define the replication server.

SBSPACETEMP

onconfig.std *value*	None

if not present	Temporary smart large objects are stored in a standard sbspace.
range of values	Up to 128 characters. SBSPACETEMP must be unique, begin with a letter or underscore, and contain only letters, digits, underscores, or **$** characters.
takes effect	When shared memory is reinitialized
utilities	**onspaces**
refer to	The following material: • "Create an Sbspace or Temporary Sbspace" on page 13-9 • "SBSPACENAME" on page 1-68 • Temporary sbspaces, in the chapter on data storage in the *IBM Informix Administrator's Guide* • Creating a temporary sbspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide* • Using temporary smart large objects, in the *IBM Informix DataBlade API Programmer's Guide*

SBSPACETEMP specifies the name of the default temporary sbspace for storing temporary smart large objects without metadata or user-data logging. If you store temporary smart large objects in a standard sbspace, the metadata is logged.

SECURESTATG

onconfig.std *value*	1
possible values	0 = All database administrators can run onstat commands to view running SQL statements. 1 = Users with GRANT DBA access to a database cannot run onstat commands to view running SQL statements.
takes effect	When the database server is shut down and restarted
refer to	*IBM Informix Administrator's Guide*

The onstat commands that show the SQL statement text that is executing on a session are normally restricted to DBA users. To remove this restriction, set the UNSECURE_ONSTAT configuration parameter to 1. Onstat commands that show SQL statements include the onstat -ses, onstat -stm, onstat -ssc, and onstat -sql.

SECURITY_LOCALCONNECTION

onconfig.std *value*	1
range of values	0 to 2 0 = no security checking occurs 1 = Dynamic Server checks whether the ID of the user who is running the program matches the ID of the user who is trying to connect to the database.

	2 = same as 1, plus Dynamic Server retrieves the peer port number from the network API and verifies that the connection is coming from the client program. You can only specify 2 if your system has DOCTCP or IPCSTR network protocols.
range of values	0 through 2
takes effect	When the database server is shut down and restarted
refer to	Role of the SERVERNUM configuration parameter, in the multiple-residency chapter of the *IBM Informix Administrator's Guide*

SECURITY_LOCALCONNECTION lets you verify security on local connections by verifying that the ID of the local user who is running a program is the same ID of the user who is trying to access the database.

SERVERNUM

onconfig.std *value*	0
range of values	0 through 255
takes effect	When the database server is shut down and restarted
refer to	Role of the SERVERNUM configuration parameter, in the multiple-residency chapter of the *IBM Informix Administrator's Guide*

SERVERNUM specifies a relative location in shared memory. The value that you choose must be unique for each database server on your local computer. The value does not need to be unique on your network. Because the value 0 is included in the **onconfig.std** file, it is suggested that you choose a value other than 0 to avoid inadvertent duplication of SERVERNUM.

SHMADD

onconfig.std ***value***	8192
range of values	32-bit platforms: 1024 through 524288 64-bit platforms: 1024 through 4294967296
units	Kilobytes
takes effect	When the database server is shut down and restarted
utilities	**onstat -g seg** (Use the **onstat -g seg** command to display the number of shared-memory segments that the database server is currently using. For more information, see page 14-20)
refer to	The following material in the *IBM Informix Administrator's Guide*: • Virtual portion of shared memory, in the shared-memory chapter • Monitoring shared-memory segments with **onstat -g seg**, in the managing memory chapter

SHMADD specifies the size of a segment that is dynamically added to the virtual portion of shared memory.

It is more efficient to add memory in large segments, but wasteful if the added memory is not used. Also, the operating system might require you to add memory in a few large segments rather than many small segments.

The following table contains recommendations for setting the initial value of SHMADD.

Amount of Physical Memory	Recommended SHMADD Value
Less than 256 megabytes	8192
Between 256 megabytes and 512 megabytes	16,384
Greater than 512 megabytes	32,768

SHMBASE

onconfig.std *value*	On UNIX: Platform dependent On Windows: `0xC000000L`
units	Address
range of values	Positive integers
takes effect	When the database server is shut down and restarted
utilities	To see the shared-memory segment addresses, use the **onstat -g seg** command.
refer to	Setting operating-system shared-memory configuration parameters, in the chapter on managing shared memory in the *IBM Informix Administrator's Guide*

SHMBASE specifies the base address where shared memory is attached to the memory space of a virtual processor. The addresses of the shared-memory segments start at the SHMBASE value and grow until the upper-bound limit, which is platform specific.

Do not change the value of SHMBASE. The **onconfig.std** value for SHMBASE depends on the platform and whether the processor is 32-bit or 64-bit. For information on which SHMBASE value to use, see the machine notes.

SHMTOTAL

onconfig.std *value*	`0`
units	Kilobytes
range of values	Integer greater than or equal to 1
takes effect	When the database server is shut down and restarted
refer to	How much shared memory the database server needs, in the shared-memory chapter of the *IBM Informix Administrator's Guide*

SHMTOTAL specifies the total amount of shared memory (resident, virtual, communications, and virtual extension portions) to be used by the database server for all memory allocations. The **onconfig.std** value of `0` implies that no limit on memory allocation is stipulated.

SHMTOTAL enables you to limit the demand for memory that the database server can place on your system. However, applications might fail if the database server requires more memory than the limit imposed by SHMTOTAL. When this situation occurs, the database server writes the following message in the message log:

```
size of resident + virtual segments xx + yy > zz total allowed by
configuration parameter SHMTOTAL
```

This message includes the following values.

Value	Description
xx	Current size of resident segments
yy	Current size of virtual segments
zz	Total shared memory required

UNIX Only

Set the operating-system parameters for maximum shared-memory segment size, typically SHMMAX, SHMSIZE, or SHMALL, to the total size that your database server configuration requires. For information on the amount of shared memory that your operating system allows, see the machine notes.

End of UNIX Only

SHMVIRTSIZE

onconfig.std *value*	`8000` on UNIX and `8192` on Windows
if not present	If SHMADD is present: `SHMADD` If SHMADD is not present: 8
units	Kilobytes
range of values	32-bit platforms: Positive integer with a maximum value of 2 gigabytes 64-bit platforms: Positive integer with a maximum value of 4 terabytes The maximum value might be less on some platforms due to operating-system limitations. For the actual maximum value for your UNIX platform, see the machine notes.
takes effect	When the database server is shut down and restarted
utilities	**onstat -g seg** (see page 14-20)
refer to	The following material: • Virtual portion of shared memory, in the shared-memory chapter of the *IBM Informix Administrator's Guide*

- Chapter on configuration effects on memory utilization, in your *IBM Informix Performance Guide*

SHMVIRTSIZE specifies the initial size of a virtual shared-memory segment. Use the following algorithm to determine the size of the virtual portion of shared memory:

```
shmvirtsize = fixed overhead + shared structures +
              mncs * private structures) + other buffers
```

This algorithm includes the following values.

Value	Description
fixed_overhead	Global pool + thread pool after booting (partially dependent on the number of virtual processors)
shared_structures	AIO vectors + sort memory + dbspace backup buffers + dictionary size + size of stored-procedure cache + histogram pool + other pools (See the **onstat** display.)
mncs	Maximum number of concurrent sessions
private_structures	Stack (generally 32 kilobytes but dependent on recursion in SPL routines and triggers) + heap (about 30 kilobytes) + session-control-block structures

If messages in the message file indicate that the database server is adding segments to the virtual portion of shared memory for you, add the amount that these messages indicate to the value of SHMVIRTSIZE. It is recommended that you initially create a virtual portion of shared memory of a size that is more than sufficient for your daily processing, if possible.

Use the **onstat -g seg** command to determine peak usage and lower the value of SHMVIRTSIZE accordingly.

SINGLE_CPU_VP

onconfig.std *value*	`0`
range of values	`0` = running with multiple CPU VPs Any nonzero value = running with one CPU VP
takes effect	When the database server is shut down and restarted
refer to	Running on a single-processor computer, in the chapter on virtual processors in the *IBM Informix Administrator's Guide*

SINGLE_CPU_VP specifies whether or not the database server is running with only one CPU virtual processor.

Setting SINGLE_CPU_VP to nonzero allows the database server to use optimized code based on the knowledge that only one CPU virtual processor is running. It enables the database server to bypass many of the mutex calls that it must use when it runs multiple CPU virtual processors.

it is strongly recommended that you set this parameter when the database server will run only one CPU virtual processor. Depending on the application and workload, setting this parameter can improve performance by up to 10 percent.

If you set SINGLE_CPU_VP to nonzero and try to add a CPU virtual processor, you receive one of the following messages:

```
onmode: failed when trying to change the number of classname VPs by n.
onmode: failed when trying to change the number of cpu virtual processors by n.
```

If you set SINGLE_CPU_VP to nonzero and then attempt to bring up the database server with VPCLASS *cpu,num* set to a value greater than 1, you receive the following error message, and the database server initialization fails:

```
Cannot have SINGLE_CPU_VP non-zero and CPU VPs greater than 1.
```

User-Defined VP Classes and SINGLE_CPU_VP

Important: Dynamic Server treats user-defined virtual-processor classes as if they were CPU virtual processors. Thus, if you set *SINGLE_CPU_VP* to nonzero, you cannot create any user-defined virtual-processor classes.

If you set this parameter to nonzero and then attempt to bring up the database server with the VPCLASS *cpu* value for *num* set to a value greater than 1, you receive the following error message, and the database server initialization fails:

```
Cannot have SINGLE_CPU_VP non-zero and CPU VPs greater than 1.
```

If you set this parameter to nonzero and then attempt to bring up the database server with a user-defined VPCLASS, you receive the following error message, and the database server initialization fails:

```
oninit: Cannot have SINGLE_CPU_VP non-zero and user-defined VP classes
```

STACKSIZE

onconfig.std *value*	32 for 32-bit database servers 64 for 64-bit database servers
units	Kilobytes
range of values	32 through limit determined by the database server configuration and the amount of memory available
takes effect	When the database server is shut down and restarted
refer to	The following material: • Stacks, in the chapter on virtual processors in the *IBM Informix Administrator's Guide* • CREATE FUNCTION statement, in the *IBM Informix Guide to SQL: Syntax*

The STACKSIZE parameter specifies the stack size for the database server user threads. The value of STACKSIZE does not have an upper limit, but setting a value that is too large wastes virtual memory space and can cause swap-space problems.

For 32-bit platforms, the default STACKSIZE value of 32 kilobytes is sufficient for nonrecursive database activity. For 64-bit platforms, the recommended STACKSIZE value is 64 kilobytes. When the database server performs recursive database tasks, as in some SPL routines, for example, it checks for the possibility of stack-size overflow and automatically expands the stack.

User threads execute user-defined routines. To increase the stack size for a particular routine, use the **stack** modifier on the CREATE FUNCTION statement.

Warning: Setting the value of *STACKSIZE* too low can cause stack overflow, the result of which is undefined but usually undesirable.

STAGEBLOB

onconfig.std *value*	None
range of values	Up to 128 characters. STAGEBLOB must be unique, begin with a letter or underscore, and contain only digits, letters, underscores, or **$** characters.
takes effect	When the database server is shut down and restarted
refer to	*IBM Informix Optical Subsystem Guide*

Use this parameter only if you are storing TEXT or BYTE data on optical storage with the Optical Subsystem. This parameter has no effect on ordinary blobspaces or sbspaces.

STAGEBLOB is the blobspace name for the area where the Optical Subsystem stages TEXT and BYTE data that is destined for storage on optical disk.

STMT_CACHE

onconfig.std *value*	None (this parameter is not in **onconfig.std**)
if not present	`0`
range of values	`0`, `1`, or `2`
takes effect	When the database server is shut down and restarted
utilities	**onmode -e**
refer to	The following material: • "Change Usage of the SQL Statement Cache" on page 10-17 • Improving query performance, in the *IBM Informix Performance Guide*

STMT_CACHE determines whether the database server uses the SQL statement cache. You can enable the SQL statement cache in one of two modes:

- Always use the SQL statement cache unless a user explicitly specifies not to use it. Set the STMT_CACHE configuration parameter to `2` or **onmode -e** `ON`.
- Use the SQL statement cache only when a user explicitly specifies to use it. Set the STMT_CACHE configuration parameter to `1` or **onmode -e** `ENABLE`.

The following table describes the possible values.

Possible Value **Meaning**

0	SQL statement cache not used (equivalent to **onmode -e** `OFF`).
1	SQL statement cache enabled, but user sessions do not use the cache. Users use the cache only if they set the environment variable **STMT_CACHE** to 1 or execute the SQL statement SET STATEMENT CACHE ON.
2	SQL statement cache turned on. All statements are cached. To turn off statement caching, set the environment variable **STMT_CACHE** to `0` or execute the SQL statement SET STATEMENT CACHE OFF.

STMT_CACHE_HITS

onconfig.std *value*	None (this parameter is not in **onconfig.std**)
if not present	`0`
units	Integer
range of values	Any value greater than or equal to `0`
takes effect	When the database server is shut down and restarted
utilities	**onmode -W STMT_CACHE_HITS** **onstat** (See "The onstat -g ssc Option" on page 14-72.)
refer to	The following material: • "Change Settings for the SQL Statement Cache" on page 10-18 • Improving query performance, in the *IBM Informix Performance Guide*

STMT_CACHE_HITS specifies the number of hits (references) to a statement before it is fully inserted in the SQL statement cache. The following table describes the possible values.

Value	Meaning
`0`	Fully insert all qualified statements in the SQL statement cache.
`>0`	The first time a user issues a unique statement, the database server inserts a *key-only* entry in the cache that identifies the statement. Subsequent identical statements increment the hit count of the *key-only* cache entry. When the hit count of the *key-only* cache entry reaches the specified number of hits, the database server fully inserts the statement in the cache. Set *hits* to 1 or more to exclude ad hoc queries from entering the cache.

STMT_CACHE_NOLIMIT

onconfig.std *value*	None (this parameter is not in **onconfig.std**)

if not present	1
range of values	0 or 1
takes effect	When the database server is shut down and restarted
utilities	**onmode -W STMT_CACHE_NOLIMIT** **onstat** (See "The onstat -g ssc Option" on page 14-72.)
refer to	The following material: • "Change Settings for the SQL Statement Cache" on page 10-18 • Improving query performance, in the *IBM Informix Performance Guide*

STMT_CACHE_NOLIMIT controls whether to insert qualified statements into the SQL statement cache after its size is greater than the STMT_CACHE_SIZE value. The following table describes the possible values.

Value	Meaning
0	Prevents statements from being inserted in the cache when its size is greater than the value of STMT_CACHE_SIZE. The cache can grow beyond the size limit if most of the statements in the cache are currently in use, because the cache cleaning cannot catch up with the insert rate. If you are concerned about memory usage, turn off STMT_CACHE_NOLIMIT to prevent the database server from allocating a large amount of memory for the cache.
1	Always insert statements in the SQL statement cache regardless of the cache size.

STMT_CACHE_NUMPOOL

onconfig.std *value*	None (this parameter is not in **onconfig.std**)
if not present	1
units	Positive integer
range of values	1 to 256
takes effect	When the database server is shut down and restarted
refer to	Improving query performance, in the *IBM Informix Performance Guide*

STMT_CACHE_NUMPOOL specifies the number of memory pools for the SQL statement cache. To obtain information about these memory pools, use **onstat -g ssc pool**.

Because the database server does not insert not all statements that allocate memory from the memory pools in the cache, the cache size might be smaller than the total size of the memory pools.

STMT_CACHE_SIZE

onconfig.std *value*	None (this parameter is not in **onconfig.std**)
default size of SQL statement cache	512 kilobytes (524288 bytes)
units	Kilobytes
range of values	Positive integer
takes effect	When the database server is shut down and restarted
utilities	**onmode -W STMT_CACHE_SIZE** **onstat-g ssc** (**Maxsize** field)
refer to	The following material: • "Change Settings for the SQL Statement Cache" on page 10-18 • Improving query performance, in the *IBM Informix Performance Guide*

The STMT_CACHE_SIZE configuration parameter specifies the size of the SQL statement cache in kilobytes. The new cache size takes effect the next time a statement is added to the cache.

SYSALARMPROGRAM

onconfig.std *value*	On UNIX: **/usr/informix/etc/evidence.sh** On Windows: **%INFORMIXDIR%\etc\evidence.bat**
range of values	Pathname
takes effect	When the database server is shut down and restarted
utilities	None
refer to	None

Set SYSALARMPROGRAM to the full pathname of the **evidence.sh** script. The database server executes **evidence.sh** when a database server failure occurs. Technical Support uses the output from the **evidence.sh** script to diagnose the cause of a database server failure.

On Windows, you must enable command extensions for **evidence.bat** to successfully complete. You can enable and disable the extensions for the Command Prompt you are working in by issuing the following commands:

- Enable: `cmd /x`
- Disable: `cmd /y`

You can also enable and disable command extensions from the Windows 2000 or Windows XP registry:

Table 1-1. Enabling command extensions from the Windows registry

Attribute	Value
Hive	HKEY_CURRENT_USER

Table 1-1. Enabling command extensions from the Windows registry (continued)

Attribute	Value
Key	Software\Microsoft\Command Processor
Name	EnableExtensions
Type	REG_DWORD
Values	0 (disable), 1 (enable)

SYSSBSPACENAME

onconfig.std *value*	None
if not present	0
range of values	Up to 128 characters. SYSSBSPACENAME must be unique, begin with a letter or underscore, and contain only digits, letters, underscores, or **$** characters.
takes effect	When disk is initialized (destroys all data)
utilities	**onspaces**
refer to	The following material:

- "Create an Sbspace or Temporary Sbspace" on page 13-9
- "Sbspace Structure" on page 3-23
- Updating statistics, in the chapter on individual query performance in your *IBM Informix Performance Guide*
- Sbspace characteristics, in the chapter on configuration effects on I/O in your *IBM Informix Performance Guide*
- Writing user-defined statistics, in the performance chapter in *IBM Informix User-Defined Routines and Data Types Developer's Guide*
- Providing statistics data for a column, in the *IBM Informix DataBlade API Programmer's Guide*
- "SBSPACENAME" on page 1-68 (specifies the name of the default sbspace)

SYSSBSPACENAME specifies the name of the sbspace in which the database server stores statistics that the UPDATE STATISTICS statement collects for certain user-defined data types. Normally, the database server stores statistics in the **sysdistrib** system catalog table.

Because the data distributions for user-defined data types can be large, you have the option to store them in an sbspace instead of in the **sysdistrib** system catalog table. If you store the data distributions in an sbspace, use DataBlade API or ESQL/C functions to examine the statistics.

Even though you specify an sbspace with the SYSSBSPACENAME parameter, you must create the sbspace with the **-c -S** option of the **onspaces** utility before you can use it. The database server validates the name of this sbspace when one of the following occurs:

- The database server attempts to write data distributions of the multirepresentational type to SYSSBSPACENAME when it executes the UPDATE STATISTICS statement with the MEDIUM or HIGH keywords.
- The database server attempts to delete data distributions of the multirepresentational type to SYSSBSPACENAME when it executes the UPDATE STATISTICS statement with the DROP DISTRIBUTIONS keywords.

Although you can store smart large objects in the sbspace specified in SYSSBSPACENAME, keeping the distribution statistics and smart large objects in separate sbspaces is recommended because:

- You avoid disk contention when queries are accessing smart large objects and the optimizer is using the distributions to determine a query plan.
- Disk space takes longer to fill up when each sbspace is used for a different purpose.

TAPEBLK

onconfig.std *value*	32
units	Kilobytes
range of values	Values greater than `pagesize/1024` To obtain the page size, see the commands listed in "System Page Size" on page 1-18
takes effect	For **ontape**: when you execute **ontape** For **onload** and **onunload**: when the database server is shut down and restarted
refer to	The following material: • Using **onload** and **onunload**, in the *IBM Informix Migration Guide* • Using **ontape**, in the *IBM Informix Backup and Restore Guide* • "LTAPEBLK" on page 1-46

TAPEBLK specifies the block size of the device to which **ontape** writes during a storage-space backup. TAPEBLK also specifies the default block size of the device to which data is loaded or unloaded when you use the **onload** or **onunload** utilities. If you are using **onload** or **onunload**, you can specify a different block size on the command line.

The database server does not check the tape device when you specify the block size. Verify that the TAPEBLK tape device can read the block size that you specify. If not, you might not able to read from the tape.

TAPEDEV

onconfig.std *value*	On UNIX: **/dev/tapedev** On Windows: **\\.\TAPE0**
if not present	On UNIX: **/dev/null**

units	Pathname
takes effect	For **ontape**: when you execute **ontape** For **onload** and **onunload**: when the database server is shut down and restarted
refer to	The following material: • Using **onload** and **onunload**, in the *IBM Informix Migration Guide* • Using **ontape**, in the *IBM Informix Backup and Restore Guide* • "LTAPEDEV" on page 1-47

TAPEDEV specifies the device to which **ontape** backs up storage spaces. TAPEDEV also specifies the default device to which data is loaded or unloaded when you use the **onload** or **onunload** utilities. In Dynamic Server 10.0 and later, you can set TAPEDEV to `STDIO` to direct back up and restore operations to standard I/O instead of to a device.

If you change the tape device, verify that TAPEBLK and TAPESIZE are correct for the new device.

Using Symbolic Links and a Remote Device (UNIX)

TAPEDEV can be a symbolic link, enabling you to switch between tape devices without changing the pathname that TAPEDEV specifies.

Use the following syntax to specify a tape device attached to another host computer:

```
host_machine_name:tape_device_pathname
```

The following example specifies a tape device on the host computer **kyoto**:

```
kyoto:/dev/rmt01
```

Rewinding Tape Devices Before Opening and on Closing

The tape device that TAPEDEV specifies must perform a rewind before it opens and when it closes. The database server requires this action because of a series of checks that it performs before it writes to a tape.

When the database server attempts to write to any tape other than the first tape in a multivolume dbspace or logical-log backup, the database server first reads the tape header to make sure that the tape is available for use. Then the device is closed and reopened. The database server assumes the tape was rewound when it closed, and the database server begins to write.

Whenever the database server attempts to read a tape, it first reads the header and looks for the correct information. The database server does not find the correct header information at the start of the tape if the tape device did not rewind when it closed during the write process.

TAPESIZE

onconfig.std *value*	`10,240`
units	Kilobytes
range of values	Positive integers

takes effect	For **ontape**: when you execute **ontape** For **onload** and **onunload**: when the database server is shut down and restarted
refer to	The following material: • Using **onload** and **onunload**, in the *IBM Informix Migration Guide* • Using **ontape**, in the *IBM Informix Backup and Restore Guide* • "LTAPESIZE" on page 1-48

Note: Tape size is irrelevant if TAPEDEV is set to `STDIO`.

The TAPESIZE parameter specifies the size of the device to which **ontape** backs up storage spaces. TAPESIZE also specifies the size of the default device to which data is loaded or unloaded when you use **onload** or **onunload**. If you are using **onload** or **onunload**, you can specify a different tape size on the command line. If you want to use the full physical capacity of a tape, set TAPESIZE to `0`.

TBLSPACE_STATS

onconfig.std *value*	`1`
if not present	`1`
units	Integer
range of values	`0` or `1`
takes effect	When the database server is shut down and restarted

The TBLSPACE_STATS configuration parameter turns on and off the collection of tblspace statistics. Use **onstat -g ppf** to list tblspace statistics.

To turn off the collection of tblspace statistics, set TBLSPACE_STATS to `0`. When TBLSPACE_STATS is set to `0`, **onstat -g ppf** displays "partition profiles disabled." To turn on the collection of tblspace statistics, set TBLSPACE_STATS to `1`.

TBLTBLFIRST

onconfig.std *value*	`0`
units	Kilobytes in multiples of page size
range of values	From the equivalent of 250 pages specified in kilobytes to the size of the first chunk minus the space needed for any system objects.
takes effect	When the database server is initialized

Specifies the first extent size of tblspace **tblspace** in the root dbspace. You might want to specify first and next extent sizes to reduce the number of tblspace **tblspace** extents and reduce the frequency of situations when you need to place the tblspace **tblspace** extents in non-primary chunks. (A primary chunk is the initial chunk in a dbspace.) For more information, see specifying first and next extent size in the chapter on managing dbspaces in the *IBM Informix Administrator's Guide*.

You can use **oncheck -pt** and **oncheck -pT** to show the first and next extent sizes of a tblspace **tblspace**. For more information about the **oncheck** utility, see “Display Tblspaces for a Table or Fragment with -pt and -pT” on page 6-15.

If you want to configure the first extent for a non-root dbspace, see information about the **onspaces** utility in Chapter 13, “Managing Storage Spaces with the onspaces Utility,” on page 13-1.

TBLTBLNEXT

onconfig.std *value*	`0`
units	Kilobytes
range of values	From equivalent of 4 pages specified in kilobytes to the maximum chunk size minus three pages
takes effect	When the database server is initialized

Specifies the next extent size of tblspace **tblspace** in the root dbspace.

If there is not enough space for a next extent in the primary chunk, the extent is allocated from another chunk. If the specified space is not available, the closest available space is allocated. For more information on configuring extent sizes in tblspace **tblspace**, see “TBLTBLFIRST” on page 1-83.

TXTIMEOUT

onconfig.std *value*	`300`
units	Seconds
range of values	Positive integers
takes effect	When the database server is shut down and restarted
refer to	How the two-phase commit protocol handles failures, in the chapter on multiphase commit protocols in the *IBM Informix Administrator's Guide*

TXTIMEOUT specifies the amount of time that a participant in a two-phase commit waits before it initiates participant recovery.

This parameter is used only for distributed queries that involve a remote database server. Nondistributed queries do not use this parameter.

USEOSTIME

onconfig.std *value*	`0`
range of values	`0` = Off 1 = On
takes effect	During initialization
refer to	The following material: • Your *IBM Informix Performance Guide*

- Using the CURRENT function to return a datetime value, in the *IBM Informix Guide to SQL: Syntax*

Setting USEOSTIME to 1 specifies that the database server is to use subsecond precision when it obtains the current time from the operating system for SQL statements. The following example shows subseconds in a datetime value:

```
2001-09-29 12:50:04.612
```

If subsecond precision is not needed, the database server retrieves the current time from the operating system once per second, making the precision of time for client applications one second. If you set USEOSTIME to `0`, the current function returns a zero (`.000`) for the year to fraction field.

When the host computer for the database server has a clock with subsecond precision, applications that depend on subsecond accuracy for their SQL statements should set USEOSTIME to 1.

Systems that run with USEOSTIME set to nonzero notice a performance degradation of up to 4 to 5 percent compared to running with USEOSTIME turned off.

This setting does not affect any calls regarding the time from application programs to Informix embedded-language library functions.

VPCLASS

onconfig.std *value*
: None

syntax
: *classname*, *options*

The *classname* variable is required. Unlike most configuration parameters, VPCLASS has several option fields that can appear in any order, separated by commas. You cannot use any white space in the fields. VPCLASS has the following options:

num=*num_VPs*
max=*max_VPs*
aff=*affinity*
noage
noyield

For more information about using these options, refer to the individual discussions later in this section.

range of values
: Up to 128 characters. VPCLASS must be unique, begin with a letter or underscore, and contain only digits, letters, underscores, or **$** characters.

takes effect
: When the database server is shut down and restarted

utilities
: **onmode -p** (to add or delete VP classes)

refer to
: The following material:

- Specifying user-defined classes of virtual processors, in the chapter on virtual processors in the *IBM Informix Administrator's Guide*
- Specifying a nonyielding user-defined virtual processor (noyield option), in the chapter on virtual processors in the *IBM Informix Administrator's Guide*
- Using **onmode -p** in "Add or Remove Virtual Processors" on page 10-12
- "Using the noyield Option" on page 1-88
- *IBM Informix User-Defined Routines and Data Types Developer's Guide*
- *J/Foundation Developer's Guide*

The VPCLASS parameter allows you to designate a class of virtual processors (VPs), create a user-defined VP, and specify the following information for it:
- The number of virtual processors that the database server should start initially
- The maximum number of virtual processors allowed for this class
- The assignment of virtual processors to CPUs if processor affinity is available
- The disabling of priority aging by the operating system if the operating system implements priority aging

You can put several VPCLASS parameter definitions in your ONCONFIG file. Each VPCLASS parameter describes one class of virtual processors. Put each definition on a separate line, as in the following example:

```
VPCLASS cpu,num=8,aff=0-7,noage
VPCLASS new,num=0
```

Default Values for the VPCLASS Options

The following table shows the defaults and value ranges for the VPCLASS parameter options.

VPCLASS option	Class	Default Value	Range of Values
aio,num	AIO	(2 * *number_of_chunks*) or 6, whichever is greater, where *number_of_chunks* is the number of chunks allocated	1 to 10,000
cpu,num	CPU	1 if MULTIPROCESSOR is 0, 2 otherwise	1 to 10,000
num	All other classes	1	1 to 10,000
max_VPs	All	Unlimited	1 to 10,000
affinity	All	VPs are assigned to available processors in round-robin fashion.	Integers from 0 to (number of CPUs -1)
noage	All	Priority aging is in effect .	**noage** or omitted
noyield	User defined	Threads will yield.	**noyield** or omitted

Interaction of VPCLASS with Other Configuration Parameters

Using the VPCLASS parameter instead of the AFF_SPROC, AFF_NPROCS, NOAGE, NUMCPUVPS, and NUMAIOVPS parameters is required. If you use VPCLASS, you must explicitly remove other parameters from your ONCONFIG file. The following table shows the parameters that you must remove.

Parameter	Parameter to Remove
VPCLASS *cpu*	NUMCPUVPS, AFF_SPROC, AFF_NPROCS, NOAGE
VPCLASS *user-defined*	SINGLE_CPU_VP
VPCLASS *aio*	NUMAIOVPS

VPCLASS Name

The first item in the VPCLASS parameter provides the name of the virtual-processor class that you are describing. The VPCLASS name is not case sensitive.

You can define new virtual-processor classes for user-defined routines or DataBlade modules, or you can set values for a predefined virtual-processor class. The following virtual-processor classes are predefined by the database server and have specific functions:

```
adm      lio      shm
adt      msc      soc
cpu      ntk      str
jvp      opt      tli
kio      aio      pio
encrypt
```

For VP classes `tli`, `shm`, `str`, and `soc`, you must set NETTYPE configuration parameter's VP_*class* field to NET.

For example, if the VPCLASS parameter is set as:

```
VPCLASS shm,num=1
VPCLASS tli,num=1
```

then the NETTYPE parameter should be set as follows:

```
NETTYPE  ipcshm,,3,NET
NETTYPE  ipctli,,3,NET
```

For more information on the NETTYPE configuration parameter, see "NETTYPE" on page 1-54.

The following example specifies that the database server should start three virtual processors of the CPU class:

```
VPCLASS cpu,num=3
```

JAVA Language Support

The JVP option of the VPCLASS configuration parameter sets the number of Java virtual processors. This parameter is required when you use the IBM Informix JDBC Driver. On UNIX, you must define multiple Java virtual processors to execute Java user-defined routines in parallel.

End of JAVA Language Support

Creating a User-Defined Class

The VPCLASS configuration parameter also allows you to create a class of user-defined virtual processors (VPs). A user-defined class of VPs can run ill-behaved user-defined routines (UDRs).

Warning: Execution of an ill-behaved routine in the *CPU VP* can cause serious interference with the operation of the database server. In addition, the routine itself might not produce correct results.

For more information on ill-behaved UDRs, see user-defined classes of virtual processors, in the chapter on virtual processors in the *IBM Informix Administrator's Guide*.

You might want to describe a user-defined class of virtual processors to run DataBlade or user-defined routines. The following example creates the user-defined class **new**, for which the database server starts three virtual processors initially:

```
VPCLASS new,num=3
```

At a later time, you can use **onmode -p** to add virtual processors to the class. The following command adds three virtual processors to the **new** class:

```
onmode -p +3 new
```

Tip: When you create a user-defined routine or function, you use the CLASS parameter of the CREATE FUNCTION statement to assign it to a class of virtual processors. You must ensure that the name of the user-defined class agrees with the name that you assigned in the CREATE FUNCTION statement. If you try to use a function that refers to a user-defined class, that class must exist and have virtual processors assigned to it. If the class does not have any virtual processors, you receive an SQL error.

For more information on how to assign a user-defined routine to either CPU or user-defined classes of virtual processors, refer to *IBM Informix User-Defined Routines and Data Types Developer's Guide*. For more information on the syntax of the CREATE FUNCTION or CREATE PROCEDURE statement, refer to the *IBM Informix Guide to SQL: Syntax*.

Using the noyield Option

By default, the VPCLASS parameter defines a yielding VP class, which allows the C UDR to yield to other threads that need access to the user-defined VP class. A UDR can perform blocking I/O calls if it executes in a yielding user-defined VP. However, it must still yield for other threads to have access to the VP.

You can also define nonyielding user-defined VPs with the **noyield** option of VPCLASS. The **noyield** option specifies creation of a nonyielding user-defined VP class. A nonyielding user-defined VP class executes a user-defined routine in a way that gives the routine exclusive use of the virtual-processor class. In other words, user-defined routines that use a **noyield** virtual-processor class run serially. They never yield the VP to another thread.

You do not need to specify more than one VP in a nonyielding user-defined VP class, because the UDR runs on a single VP until it completes and any additional virtual processors would be idle.

Important: If your *UDR* uses global variables, only one *VP* in the user-defined virtual-processor class should be nonyielding.

The following example specifies a user-defined class of virtual processors called **new_noyield**, which runs in no-yield mode:

```
VPCLASS new_noyield,noyield,num=1
```

The **noyield** option applies only to user-defined VP classes. The database server ignores **noyield** if it is part of a VPCLASS parameter that defines a predefined VP class such as CPU, AIO, and so on.

Using the num Option

The *num* option sets the number of virtual processors of the specified class that the database server should start during initialization.

On a single-processor computer, allocate only one CPU virtual processor. On a multiprocessor computer, do not allocate more CPU and user-defined virtual processors, combined, than there are CPUs on the computer.

Use the following syntax to specify the number of virtual processors:

```
num=num_VPs
```

Specifying the Number of CPU VPs

For example, the following parameter specifies that the database server should start four virtual processors for the **cpu** class:

```
VPCLASS cpu,num=4
```

At a later time, you can use the **onmode -p** command to add virtual processors for the class.

Using the max_VPs Option

The *max*_VP*s* option specifies the maximum number of virtual processors that the database server can start for the class.

Use the following syntax to specify the number of virtual processors:

```
max=max_VPs
```

The value can be any integer greater than 0. If you omit the *max_VPs* option, the number is unlimited.

Using the affinity Option

On multiprocessor computers that support *processor affinity*, the affinity option specifies the CPUs to which the database server binds virtual processors.

The affinity option has the following two forms:

```
aff=processor_number
aff=start_range,end_range
```

In the first form, the database server binds all virtual processors in the class to the CPU numbered *processor_number*. (On a multiprocessor system, the operating system numbers the CPUs from 0 to (`number of CPUs-1`)). In the second form, the database server assigns the virtual processors of the class to processors in the range *start_range* to *end_range*, inclusive. The value *end_range* must be larger than *start_range*, and all values must be less than the total number of available CPUs.

For example, if your platform has eight CPUs, your ONCONFIG file might include the following VPCLASS entries:

```
VPCLASS     first,aff=3
VPCLASS     second,num=3,aff=5-7
VPCLASS     cpu,num=8,aff=0-7,noage
```

For more information about using processor affinity, refer to the chapter on virtual processors in the *IBM Informix Administrator's Guide*.

Chapter 2. The sysmaster Database

In This Chapter

This chapter describes the **sysmaster** database and contains reference information for the *system-monitoring interface* (SMI). It provides information on the following topics:

- What is the **sysmaster** database
- How to use SMI tables
- Descriptions of the SMI tables
- A map of the documented SMI tables

For information about the ON–Bar tables, see the *IBM Informix Backup and Restore Guide*.

The sysmaster Database

The database server creates and maintains the **sysmaster** database. It is analogous to the system catalog for databases, which is described in the *IBM Informix Guide to SQL: Reference*. Just as a system catalog for every database managed by the database server keeps track of objects and privileges in the database, a **sysmaster** database for every database server keeps track of information about the database server.

The **sysmaster** database contains the *system-monitoring interface* (SMI) tables. The SMI tables provide information about the state of the database server. You can query these tables to identify processing bottlenecks, determine resource usage, track session or database server activity, and so on. This chapter describes these tables, which are slightly different from ordinary tables.

Warning: The database server relies on information in the **sysmaster** database. Do not change any of the tables in **sysmaster** or any of the data within the tables. Such changes could cause unpredictable and debilitating results.

The database server creates the **sysmaster** database when it initializes disk space. The database server creates the database with unbuffered logging. You cannot drop the database or any of the tables in it, and you cannot turn logging off.

As user **informix** on UNIX or a member of the **Informix-Admin** group on Windows, you can create SPL routines in the **sysmaster** database. (You can also create triggers on tables within **sysmaster**, but the database server never executes those triggers.)

Joins of multiple tables in **sysmaster** might return inconsistent results because the database server does not lock the tables during a join. You can join **sysmaster** tables with tables in other databases. However, to join **sysmaster** tables with tables in a nonlogging database, first make the nonlogging database the current database.

The buildsmi Script

When you bring the database server up for the first time, it runs a script called **buildsmi**, which is in the **etc** directory. This script builds the database and tables that support SMI. The database server requires approximately 1750 free pages of logical-log space to build the **sysmaster** database.

If you receive an error message that directs you to run the **buildsmi** script, a problem probably occurred while the database server was building the SMI database, tables, and views. When you use **buildsmi**, the existing **sysmaster** database is dropped and then re-created.

The bldutil.sh Script

When you initialize the database server for the first time, it runs a script called **bldutil.sh** on UNIX or **bldutil.bat** on Windows. This script builds the **sysutils** database. If it fails, the database server creates an output file in the **tmp** directory. The output file is **bldutil.process_id** on UNIX and **bldutil.out** on Windows. The messages in this output file reflect errors that occurred during the script execution.

The System-Monitoring Interface

This section describes the SMI tables and how you access them to monitor the database server operation.

Understanding the SMI Tables

The system-monitoring interface consists of tables and pseudo-tables that the database server maintains automatically. While the SMI tables appear to the user as tables, they are not recorded on disk as normal tables are. Instead, the database server constructs the tables in memory, on demand, based on information in shared memory at that instant. When you query an SMI table, the database server reads information from these shared-memory structures. Because the database server continually updates the data in shared memory, the information that SMI provides lets you examine the current state of your database server.

The SMI tables provide information about the following topics:

- Auditing
- Disk usage
- User profiling
- Database-logging status
- Tables
- Chunks
- Chunk I/O
- Dbspaces
- Locks
- Extents
- SQL statement cache statistics
- Virtual-processor CPU usage
- System profiling

The data in the SMI tables changes dynamically as users access and modify databases that the database server manages.

Accessing SMI Tables

Any user can use SQL SELECT statements to query an SMI table, but standard users cannot execute statements other than SELECT. Attempts to do so result in permission errors. The administrator can execute SQL statements other than SELECT, but the results of such statements are unpredictable.

Dynamic Server provides the **sysadtinfo** and **sysaudit** tables. Only user **informix** on UNIX or members of the **Informix-Admin** group on Windows can query **sysadtinfo** and **sysaudit**.

You cannot use **dbschema** or **dbexport** on any of the tables in the **sysmaster** database. If you do, the database server generates the following error message:

```
Database has pseudo tables - can't build schema
```

SELECT Statements

You can use SELECT statements on SMI tables wherever you can use SELECT against ordinary tables (from DB–Access, in an SPL routine, with ESQL/C, and so on) with one restriction: you cannot (meaningfully) reference **rowid** when you query SMI tables. SELECT statements that use **rowid** do not return an error, but the results are unpredictable.

All standard SQL syntax, including joins between tables, sorting of output, and so on, works with SMI tables. For example, if you want to join an SMI table with a non-SMI table, name the SMI table with the following standard syntax:

```
sysmaster[@dbservername]:[owner.]tablename
```

Triggers and Event Alarms

Triggers based on changes to SMI tables never run. Although you can define triggers on SMI tables, triggers are activated only when an INSERT, UPDATE, or DELETE statement occurs on a table. The updates to the SMI data occur within the database server, without the use of SQL, so a trigger on an SMI table is never activated, even though the data returned by a SELECT statement indicates that it should be.

To create an event alarm, query for a particular condition at predefined intervals, and execute an SPL routine if the necessary conditions for the alarm are met.

SPL and SMI Tables

You can access SMI tables from within a SPL routine. When you reference SMI tables, use the same syntax that you use to reference a standard table.

Locking and SMI Tables

The information in the SMI tables changes based on the database server activity. However, the database server does not update the information using SQL statements. When you use SMI tables with an isolation level that locks objects, it prevents other users from accessing the object, but it does not prevent the data from changing. In this sense, all the SMI tables have a permanent Dirty Read isolation level.

The System-Monitoring Interface Tables

The database server supports the following SMI tables.

Table	Description	Reference
sysadtinfo	Auditing configuration information	page 2-5
sysaudit	Auditing event masks	page 2-5
syschkio	Chunk I/O statistics	page 2-6
syschunks	Chunk information	page 2-6
sysconfig	Configuration information	page 2-7
sysdatabases	Database information	page 2-8
sysdbslocale	Locale information	page 2-8
sysdbspaces	Dbspace information	page 2-10
sysdri	Data-replication information	page 2-9
sysextents	Extent-allocation information	page 2-10
sysextspaces	External spaces information	page 2-10
syslocks	Active locks information	page 2-11
syslogs	Logical-log file information	page 2-11
sysprofile	System-profile information	page 2-12
sysptprof	Table information	page 2-13
syssesprof	Counts of various user actions	page 2-14
syssessions	Description of each user connected	page 2-15
sysseswts	User's wait time on each of several objects	page 2-16
systabnames	Database, owner, and table name for the tblspace **tblspace**	page 2-17

Table	Description	Reference
sysvpprof	User and system CPU used by each virtual processor	page 2-17

Many other tables in the **sysmaster** database are part of the system-monitoring interface but are not documented. Their schemas and column content can change from version to version. The flags_text table now contains more rows. To view the new rows you must first drop and then recreate the sysmaster database.

The sysutils Tables

ON–Bar uses the following tables in the **sysutils** database. For more information, see the *IBM Informix Backup and Restore Guide*.

Table	Description
bar_action	Lists all backup and restore actions that are attempted against an object, except during a cold restore. Use the information in this table to track backup and restore history.
bar_instance	Writes a record to this table for each successful backup. ON–Bar might later use the information for a restore operation.
bar_object	Describes each backup object. This table provides a list of all storage spaces and logical logs from each database server for which at least one backup attempt was made.
bar_server	Lists the database servers in an installation. This table is used to ensure that backup objects are returned to their proper places during a restore.

sysadtinfo

The **sysadtinfo** table contains information about the auditing configuration for the database server. For more information, see your *IBM Informix Trusted Facility Guide*. You must be user **informix** or user **root** on UNIX or a member of the **Informix-Admi**n group on Windows to retrieve information from the sysadtinfo table.

Column	Type	Description
adtmode	integer	If auditing is on or off: • 0 For off • 1 For on
adterr	integer	Action on errors: • 0 To continually retry audit writes until they succeed. Processing for the thread that generated the error stops. • 1 To write all failed audit writes to the message log and continue processing.
adtsize	integer	Maximum size of an audit file
adtpath	char(256)	Directory where audit files are written
adtfile	integer	Number of the audit file

sysaudit

For each defined audit mask (that is, for each *username*), the **sysaudit** table contains flags that represent the database events that generate audit records. The **success** and **failure** columns represent the bitmasks that compose the audit masks. If a bit is set in both the **success** the and **failure** columns, the corresponding event generates an audit record whether or not the event succeeded.

You must be user **informix** or **user** root on UNIX or a member of the **Informix-Admin** group on Windows to retrieve information from the **sysaudit** table.

Use the **onaudit** utility to list or modify an audit mask. For information about **onaudit** and auditing, see your *IBM Informix Trusted Facility Guide*.

Column	Type	Description
username	char(32)	Name of the mask
succ1	integer	Bitmask of the audit mask for success
succ2	integer	Bitmask of the audit mask for success
succ3	integer	Bitmask of the audit mask for success
succ4	integer	Bitmask of the audit mask for success
succ5	integer	Bitmask of the audit mask for success
fail1	integer	Bitmask of the audit mask for failure
fail2	integer	Bitmask of the audit mask for failure
fail3	integer	Bitmask of the audit mask for failure
fail4	integer	Bitmask of the audit mask for failure
fail5	integer	Bitmask of the audit mask for failure

syschkio

The **syschkio** table provides I/O statistics for individual chunks that the database server manages.

Column	Type	Description
chunknum	smallint	Chunk number
reads	integer	Number of physical reads
pagesread	integer	Number of pages read
writes	integer	Number of physical writes
pageswritten	integer	Number of pages written
mreads	integer	Number of physical reads (mirror)
mpagesread	integer	Number of pages read (mirror)
mwrites	integer	Number of physical writes (mirror)
mpageswritten	integer	Number of pages written (mirror)

syschunks

The **syschunks** table describes each of the chunks that the database server manages. In the **flags** and **mflags** columns, each bit position represents a separate flag. Thus, it might be easier to read values in the **flags** and **mflags** columns if the values are returned using the HEX function.

<table>
<tr><th>Column</th><th>Type</th><th colspan="3">Description</th></tr>
<tr><td>chknum</td><td>smallint</td><td colspan="3">Chunk number</td></tr>
<tr><td>dbsnum</td><td>smallint</td><td colspan="3">Dbspace number</td></tr>
<tr><td>nxchknum</td><td>smallint</td><td colspan="3">Number of the next chunk in this dbspace</td></tr>
<tr><td>chksize</td><td>integer</td><td colspan="3">Number of pages in this chunk</td></tr>
<tr><td>offset</td><td>integer</td><td colspan="3">Page offset of the chunk in its device or path</td></tr>
<tr><td>pagesize</td><td>integer</td><td colspan="3">Page size (in units of system default page size)</td></tr>
<tr><td>nfree</td><td>integer</td><td colspan="3">Number of free pages in the chunk (in units of system default page size)</td></tr>
<tr><td>is_offline</td><td>integer</td><td colspan="3">1 If the chunk is offline, 0 if not</td></tr>
<tr><td>is_recovering</td><td>integer</td><td colspan="3">1 If the chunk is being recovered, 0 if not</td></tr>
<tr><td>is_blobchunk</td><td>integer</td><td colspan="3">1 If the chunk is in a blobspace, 0 if not</td></tr>
<tr><td>is_sbchunk</td><td>integer</td><td colspan="3">1 If the chunk is a sbspace, 0 if not</td></tr>
<tr><td>is_inconsistent</td><td>integer</td><td colspan="3">1 If the chunk is undergoing logical restore, 0 if not</td></tr>
<tr><td rowspan="13">flags</td><td rowspan="13">smallint</td><td>Flags</td><td>Hexadecimal</td><td>Meaning</td></tr>
<tr><td>16</td><td>0x0010</td><td>Chunk is a mirrored chunk</td></tr>
<tr><td>32</td><td>0x0020</td><td>Chunk is in offline mode</td></tr>
<tr><td>64</td><td>0x0040</td><td>Chunk is in online mode</td></tr>
<tr><td>128</td><td>0x0080</td><td>Chunk is in recovery mode</td></tr>
<tr><td>256</td><td>0x0100</td><td>Chunk has just been mirrored</td></tr>
<tr><td>512</td><td>0x0200</td><td>Chunk is part of a blobspace</td></tr>
<tr><td>1024</td><td>0x0400</td><td>Chunk is being dropped</td></tr>
<tr><td>2048</td><td>0x0800</td><td>Chunk is part of an optical stageblob</td></tr>
<tr><td>4096</td><td>0x1000</td><td>Chunk is inconsistent</td></tr>
<tr><td>16384</td><td>0x4000</td><td>Chunk contains temporary log space</td></tr>
<tr><td>32768</td><td>0x8000</td><td>Chunk was added during roll forward</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>fname</td><td>char(256)</td><td colspan="3">Pathname for the file or device of this chunk</td></tr>
<tr><td>mfname</td><td>char(256)</td><td colspan="3">Pathname for the file or device of the mirrored chunk, if any</td></tr>
<tr><td>moffset</td><td>integer</td><td colspan="3">Page offset of the mirrored chunk</td></tr>
<tr><td>mis_offline</td><td>integer</td><td colspan="3">1 If mirror is offline, 0 if not</td></tr>
<tr><td>mis_recovering</td><td>integer</td><td colspan="3">1 If mirror is being recovered, 0 if not</td></tr>
<tr><td>mflags</td><td>smallint</td><td colspan="3">Mirrored chunk flags; values and meanings are the same as the flags column.</td></tr>
</table>

sysconfig

The **sysconfig** table describes the effective, original, and default values of the configuration parameters. For more information about the ONCONFIG file and the configuration parameters, see Chapter 1, “Configuration Parameters,” on page 1-1.

Column	Type	Description
cf_id	integer	Unique numeric identifier
cf_name	char(128)	Configuration parameter name
cf_flags	integer	Reserved for future use
cf_original	char(256)	Value in the ONCONFIG file at boot time
cf_effective	char(256)	Value currently in use
cf_default	char(256)	Value provided by the database server if no value is specified in the ONCONFIG file

sysdatabases

The **sysdatabases** table describes each database that the database server manages.

Column	Type	Description	
name	char(128)	Database name	
partnum	integer	The partition number (tblspace identifier) for the systables table for the database	
owner	char(32)	User ID of the creator of the database	
created	date	Date created	
is_logging	integer	1 If logging is active, 0 if not	
is_buff_log	integer	1 If buffered logging, 0 if not	
is_ansi	integer	1 If ANSI-compliant, 0 if not	
is_nls	integer	1 If GLS-enabled, 0 if not	
flags	smallint	Logging flags (hex values)	
		0	No logging
		1	Unbuffered logging
		2	Buffered logging
		4	ANSI-compliant database
		8	Read-only database
		10	GLS database
		20	Checking of the logging mode of **syscdr** database bypassed
		100	Changed status to buffered logging
		200	Changed status to unbuffered logging
		400	Changed status to ANSI compliant
		800	Database logging turned off
		1000	Long ID support enabled

sysdbslocale

The **sysdbslocale** table lists the locale of each database that the database server manages.

Column	Type	Description
dbs_dbsname	char(128)	Database name
dbs_collate	char(32)	The locale of the database

sysdbspaces

The **sysdbspaces** table describes each of the dbspaces that the database server manages. In the **flags** column, each bit position represents a separate flag. Thus, it might be easier to read values in the **flags** column if the values are returned using the HEX function.

Column	Type	Description		
dbsnum	smallint	Dbspace number		
name	char(128)	Dbspace name		
owner	char(32)	User ID of owner of the dbspace		
fchunk	smallint	Number of the first chunk in the dbspace		
nchunks	smallint	Number of chunks in the dbspace		
pagesize	integer	Page size		
is_mirrored	integer	1 If dbspace is mirrored, 0 if not		
is_blobspace	integer	1 If the dbspace is a blobspace, 0 if not		
is_sbspace	integer	1 If the dbspace is a sbspace, 0 if not		
is_temp	integer	1 If the dbspace is a temporary dbspace, 0 if not		
flags	smallint	Flags	Hexadecimal	Meaning
		1	0x0001	Dbspace has no mirror
		2	0x0002	Dbspace uses mirroring
		4	0x0004	Dbspace mirroring is disabled
		8	0x0008	Dbspace is newly mirrored
		16	0x0010	Space is a blobspace
		32	0x0020	Blobspace is on removable media
		64	0x0040	Blobspace is on optical media
		128	0x0080	Blobspace has been dropped.
		256	0x0100	Blobspace is an optical stageblob
		512	0x0200	Space is being recovered
		1024	0x0400	Space has been physically recovered
		2048	0x0800	Space is in logical recovery
		32768	0x8000	Space is an sbspace

sysdri

The **sysdri** table provides information about the High-Availability Data-Replication status of the database server.

Column	Type	Description
type	char(50)	High-Availability Data Replication type Possible values: • primary • secondary • standard • not initialized
state	char(50)	State of High-Availability Data Replication Possible values: • off • on • connecting • failure • read-only
name	char(128)	The name of the other database server in the High-Availability Data-Replication pair
intvl	integer	The High-Availability Data-Replication interval
timeout	integer	The High-Availability Data-Replication timeout value for this database server
lostfound	char(256)	The pathname to the lost-and-found file

sysextents

The **sysextents** table provides information about extent allocation.

Column	Type	Description
dbsname	char(128)	Database name
tabname	char(128)	Table name
chunk	integer	Chunk number
offset	integer	Number of pages into the chunk where the extent begins
size	integer	Size of the extent, in pages

sysextspaces

The sysextspaces table provides information about external spaces. Indexes for the **id** column and the **name** column allow only unique values.

Column	Type	Description
id	integer	External space ID
name	char(128)	External space name
owner	char(32)	External space owner
flags	integer	External space flags (reserved for future use)
refcnt	integer	External space reference count.
locsize	integer	Size of external space location, in bytes
location	char (256)	Location of external space

syslocks

The **syslocks** table provides information about all the currently active locks in the database server.

Column	Type	Description	
dbsname	char(128)	Database name	
tabname	char(128)	Table name	
rowidlk	integer	Real rowid, if it is an index key lock	
keynum	smallint	Key number of index key lock	
type	char(4)	Type of lock	
		B	Byte lock
		IS	Intent shared lock
		S	Shared lock
		XS	Shared key value held by a repeatable reader
		U	Update lock
		IX	Intent exclusive lock
		SIX	Shared intent exclusive lock
		X	Exclusive lock
		XR	Exclusive key value held by a repeatable reader
owner	integer	Session ID of the lock owner	
waiter	integer	Session ID of the user waiting for the lock. If more than one user is waiting, only the first session ID appears.	

syslogs

The **syslogs** table provides information about space use in logical-log files. In the **flags** column, each bit position represents a separate flag. For example, for a log file, the **flags** column can have flags set for both current log file and temporary log file. Thus, it might be easier to read values in the **flags** column if the values are returned using the HEX function.

Column	Type	Description
number	smallint	Logical-log file number
uniqid	integer	Log-file ID
size	integer	Number of pages in the log file
used	integer	Number of pages used in the log file
is_used	integer	1 If file is used, 0 if not
is_current	integer	1 If file is the current file, 0 if not
is_backed_up	integer	1 If file has been backed up, 0 if not
is_new	integer	1 If the log has been added since the last level-0 dbspace backup, 0 if not
is_archived	integer	1 If file has been placed on the backup tape, 0 if not
is_temp	integer	1 If the file is flagged as a temporary log file, 0 if not

Column	Type	Description		
flags	smallint	Flags	Hexadecimal	Meaning
		1	0x01	Log file is in use
		2	0x02	File is current log file
		4	0x04	Log file has been backed up
		8	0x08	File is newly added log file
		16	0x10	Log file has been written to dbspace backup media
		32	0x20	Log is a temporary log file

sysprofile

The **sysprofile** table contains profile information about the database server.

Column	Type	Description
name	char(13)	Name of profiled event. (See table that follows for a list of possible events.)
value	integer	Value of profiled event. (See table that follows for a list of possible events.)

The following table lists the events that, together with a corresponding value, make up the rows of the **sysprofile** table.

Events Profiled in sysprofile	Description
dskreads	Number of actual reads from disk
bufreads	Number of reads from shared memory
dskwrites	Actual number of writes to disk
bufwrites	Number of writes to shared memory
isamtot	Total number of calls
isopens	isopen calls
isstarts	isstart calls
isreads	isread calls
iswrites	iswrite calls
isrewrites	isrewrite calls
isdeletes	isdelete calls
iscommits	iscommit calls
isrollbacks	isrollback calls
ovlock	Overflow lock table
ovuser	Overflow user table
ovtrans	Overflow transaction table
latchwts	Latch request waits
bufwts	Buffer waits
lockreqs	Lock requests
lockwts	Lock waits

Events Profiled in sysprofile	Description
ckptwts	Checkpoint waits
deadlks	Deadlocks
lktouts	Deadlock time-outs
numckpts	Number checkpoints
plgpagewrites	Physical-log pages written
plgwrites	Physical-log writes
llgrecs	Logical-log records
llgpagewrites	Logical-log writes
llgwrites	Logical-log pages written
pagreads	Page reads
pagwrites	Page writes
flushes	Buffer-pool flushes
compress	Page compresses
fgwrites	Foreground writes
lruwrites	Least-recently used (LRU) writes
chunkwrites	Writes during a checkpoint
btradata	Read-ahead data pages read through index leaf node
btraidx	Read-ahead data pages read through index branch or root node
dpra	Data pages read into memory with read-ahead feature
rapgs_used	Read-ahead data pages that user used
seqscans	Sequential scans
totalsorts	Total sorts
memsorts	Sorts that fit in memory
disksorts	Sorts that did not fit in memory
maxsortspace	Maximum disk space used by a sort

sysptprof

The **sysptprof** table lists information about a tblspace. Tblspaces correspond to tables.

Profile information for a table is available only when a table is open. When the last user who has a table open closes it, the tblspace in shared memory is freed, and any profile statistics are lost.

Column	Type	Description
dbsname	char(128)	Database name
tabname	char(128)	Table name
partnum	integer	Partition (tblspace) number
lockreqs	integer	Number of lock requests
lockwts	integer	Number of lock waits
deadlks	integer	Number of deadlocks
lktouts	integer	Number of lock timeouts

Column	Type	Description
isreads	integer	Number of isreads
iswrites	integer	Number of iswrites
isrewrites	integer	Number of isrewrites
isdeletes	integer	Number of isdeletes
bufreads	integer	Number of buffer reads
bufwrites	integer	Number of buffer writes
seqscans	integer	Number of sequential scans
pagreads	integer	Number of page reads
pagwrites	integer	Number of page writes

syssesprof

The **syssesprof** table lists cumulative counts of the number of occurrences of user actions such as writes, deletes, or commits.

Column	Type	Description
sid	integer	Session ID
lockreqs	integer	Number of locks requested
locksheld	integer	Number of locks currently held
lockwts	integer	Number of times waited for a lock
deadlks	integer	Number of deadlocks detected
lktouts	smallint	Number of deadlock timeouts
logrecs	integer	Number of logical-log records written
isreads	integer	Number of reads
iswrites	integer	Number of writes
isrewrites	integer	Number of rewrites
isdeletes	integer	Number of deletes
iscommits	integer	Number of commits
isrollbacks	integer	Number of rollbacks
longtxs	integer	Number of long transactions
bufreads	integer	Number of buffer reads
bufwrites	integer	Number of buffer writes
seqscans	integer	Number of sequential scans
pagreads	integer	Number of page reads
pagwrites	integer	Number of page writes
total_sorts	integer	Number of total sorts
dsksorts	integer	Number of sorts that did not fit in memory
max_sortdiskspace	integer	Maximum space used by a sort
logspused	integer	Number of bytes of logical-log space used by current transaction of session
maxlogsp	integer	Maximum number of bytes of logical-log space ever used by the session

syssessions

The **syssessions** table provides general information on each user connected to the database server. In the **state** column, each bit position represents a separate flag. Thus, it might be easier to read values in the **state** column if the values are returned using the HEX function.

Column	Type	Description
sid	integer	Session ID
username	char(32)	User ID
uid	smallint	User ID number
pid	integer	Process ID of the client
hostname	char(16)	Hostname of client
tty	char(16)	Name of the user's **stderr** file
connected	integer	Time that user connected to the database server
feprogram	char(16)	Reserved for future use
pooladdr	integer	Session pool address
is_wlatch	integer	1 If the primary thread for the session is waiting for a latch
is_wlock	integer	1 If the primary thread for the session is waiting for a lock
is_wbuff	integer	1 If the primary thread for the session is waiting for a buffer
is_wckpt	integer	1 If the primary thread for the session is waiting for a checkpoint
is_wlogbuf	integer	1 If the primary thread for the session is waiting for a log buffer
is_wtrans	integer	1 If the primary thread for the session is waiting for a transaction
is_monitor	integer	1 If the session is a special monitoring process
is_incrit	integer	1 If the primary thread for the session is in a critical section

Column	Type	Description		
state	integer	Flags	Hexadecimal	Meaning
		1	0x00000001	User structure in use
		2	0x00000002	Waiting for a latch
		4	0x00000004	Waiting for a lock
		8	0x00000008	Waiting for a buffer
		16	0x00000010	Waiting for a checkpoint
		32	0x00000020	In a read call
		64	0x00000040	Writing logical-log file to backup tape
		128	0x00000080	ON-Monitor (UNIX)
		256	0x00000100	In a critical section
		512	0x00000200	Special daemon
		1024	0x00000400	Archiving
		2048	0x00000800	Clean up dead processes
		4096	0x00001000	Waiting for write of log buffer
		8192	0x00002000	Special buffer-flushing thread
		16384	0x00004000	Remote database server
		32768	0x00008000	Deadlock timeout used to set RS_timeout
		65536	0x00010000	Regular lock timeout
		262144	0x00040000	Waiting for a transaction
		524288	0x00080000	Primary thread for a session
		1048576	0x00100000	Thread for building indexes
		2097152	0x00200000	B-tree cleaner thread

sysseswts

The **sysseswts** table provides information on the amount of time that users wait for various database objects.

Column	Type	Description
sid	integer	Session ID

Column	Type	Description
reason	char(50)	Description of reason for wait: • Unspecified • Buffer • Lock • Asynchronous I/O • Mt yield 0 • Mt yield n • Mt yield • Checkpoint • Log I/O • Log copy • Condition • Lock mutex • Lockfree mutex • Deadlock mutex • LRUs mutex • Tblspace mutex • Log mutex • Checkpoint mutex • Mutex • Mt ready • Mt yield x • Running
numwaits	integer	Number of waits for this reason
cumtime	float	Cumulative time waited for this reason (in microseconds)
maxtime	integer	Maximum time waited during this session for this reason

systabnames

The **systabnames** table describes each table that the database server manages.

Column	Type	Description
partnum	integer	Tblspace identifier
dbsname	char(128)	Database name
owner	char(32)	User ID of owner
tabname	char(128)	Table name
collate	char(32)	Collation associated with a database that supports GLS

sysvpprof

The **sysvpprof** table lists user and system CPU time for each virtual processor.

Column	Type	Description
vpid	integer	Virtual processor ID

Column	Type	Description
class	char(50)	Type of virtual processor: • cpu • adm • lio • pio • aio • tli • soc • str • shm • opt • msc • adt
usercpu	float	Number of microseconds of user time
syscpu	float	Number of microseconds of system time

The SMI Tables Map

Figure 2-1 displays the columns in some of the SMI tables.

sysadtinfo	sysaudit	syschkio	syschunks	sysconfig	sysdatabases
adtmode	username	chunknum	chknum	cf_id	name
adterr	succ1	reads	dbsnum	cf_name	partnum
adtsize	succ2	pagesread	nxchknum	cf_flags	owner
adtpath	succ3	writes	chksize	cf_originals	created
adtfile	succ4	pageswritten	offset	cf_effective	is_logging
	succ5	mreads	nfree	cf_default	is_buff_log
	fail1	mpagesread	Is_offline		is_ansi
	fail2	mwrites	is_recovering		is_nls
	fail3	mpageswritten	is_blobchunk		flags
	fail4		is_sbchunk		
	fail5		is_inconsistent		
			flags		
			fname		
			mfname		
			moffset		
			mis_offline		
			mis_recovering		
			mflags		

Figure 2-1. Columns in the SMI tables (Part 1 of 4)

sysdbslocale
dbs_dbsname
dbs_collate

sysdbspaces
dbsnum
name
owner
fchunk
nchunks
is_mirrored
is_blobspace
is_sbspace
is_temp
flags

sysdri
type
state
name
intvl
timeout
lostfound

sysextents
dbsname
tabname
chunk
offset
size

sysextspaces
id
name
owner
flags
refcnt
locsize
location

syslocks
dbsname
tabname
rowidlk
keynum
type
owner
waiter

Figure 2-1. Columns in the SMI tables (Part 2 of 4)

syslogs
number
uniqid
size
used
is_used
is_current
is_backed_up
is_new
is_archived
is_temp
flags

sysprofile
name
value

sysptprof
dbsname
tabname
partnum
lockreqs
lockwts
deadlks
lktouts
isreads
iswrites
isrewrites
isdeletes
bufreads
bufwrites
seqscans
pagreads
pagwrites

syssesprof
sid
lockreqs
locksheld
lockwts
deadlks
lktouts
logrecs
isreads
iswrites
isrewrites
isdeletes
iscommits
isrollbacks
longtxs
bufreads
bufwrites
seqscans
pagreads
pagwrites
total_sorts
dsksorts
max_sort diskspace
logspused
maxlogsp

syssessions
sid
username
uid
pid
hostname
tty
connected
feprogram
pooladdr
is_wlatch
is_wlock
is_wbuff
is_wckpt
is_wlogbuf
is_wtrans
is_monitor
is_incrit
state

Figure 2-1. Columns in the SMI tables (Part 3 of 4)

sysseswts
sid
reason
numwaits
cumtime
maxtime

systabnames
partnum
dbsname
owner
tabname
collate

sysvpprof
vpid
class
usercpu
syscpu

Figure 2-1. Columns in the SMI tables (Part 4 of 4)

Information from onstat in the SMI Tables

To obtain information provided by the **onstat** utility, you can use SQL to query appropriate SMI tables. The following table indicates which SMI tables to query to obtain the information provided by a given **onstat** option. For descriptions of the **onstat** options, see "Monitor the Database Server Status" on page 14-3.

onstat Option	**SMI Tables to Query**	**onstat Fields *Not* in SMI Tables**
-d	sysdbspaces syschunks	address bpages
-D	sysdbspaces syschkio	
-F	sysprofile	address flusher snoozer state data
-g dri	sysdri	Last DR CKPT (id/pg)
-g glo	sysvpprof	Listing of virtual processors by class
-k	syslocks	address lklist tblsnum
-l	syslogs sysprofile	All physical-log fields (except numpages and numwrits) All logical-log buffer fields (except numrecs, numpages, and numwrits) address begin % used
-p	sysprofile	
-u	syssessions syssesprof	address wait nreads nwrites

Chapter 3. Disk Structures and Storage

In This Chapter

The database server achieves its high performance by managing its own I/O. The database server manages storage, search, and retrieval. As the database server stores data, it creates the structures it needs to search for and retrieve the data later. The database server disk structures also store and track control information needed to manage logging and backups. Database server structures contain all the information needed to ensure data consistency, both physical and logical.

Before you read this chapter, familiarize yourself with the disk-space terms and definitions in the chapter on where data is stored in the *IBM Informix Administrator's Guide*.

This chapter discusses the following topics related to disk data structures:
- Dbspace structure and storage
- Storage of simple large objects
- Sbspace structure
- Time stamps
- Database and table creation: what happens on disk

Dbspace Structure and Storage

This section explores the disk structures and storage techniques that the database server uses to store data in a dbspace.

Structure of the Root Dbspace

The ROOTNAME, ROOTOFFSET, ROOTPATH, and ROOTSIZE configuration parameters specify the size and location of the initial chunk of the root dbspace. If the root dbspace is mirrored, the MIRROROFFSET and MIRRORPATH configuration parameters specify the mirror-chunk location. For more information about these parameters, see Chapter 1, “Configuration Parameters,” on page 1-1.

As part of disk-space initialization, the database server initializes the following structures in the initial chunk of the root dbspace:
- Twelve reserved pages
- The first chunk free-list page
- The tblspace tblspace
- The database tblspace
- The physical log
- The logical-log files
- **oncheck -pe**

  For more information, see “Check the Chunk Free List with -ce and -pe” on page 6-10.

Reserved Pages

The first 12 pages of the initial chunk of the root dbspace are reserved pages. Each reserved page contains specific control and tracking information used by the database server**.**

To obtain a listing of the contents of your reserved pages, execute the command **oncheck -pr**. To also list information about the physical-log and logical-log pages, including the active physical-log pages, execute **oncheck -pR**.

Structure of a Regular Dbspace

After disk-space initialization, you can add new dbspaces. When you create a dbspace, you assign at least one chunk (either raw or cooked disk space) to the dbspace. This chunk is referred to as the initial chunk of the dbspace. Figure 3-1 on page 3-3 illustrates the structure of the initial chunk of a regular (nonroot) dbspace.

When the dbspace is first created, it contains the following structures:

- Two reserved pages
- The first chunk free-list page in the chunk
- The tblspace **tblspace** for this dbspace
- Unused pages

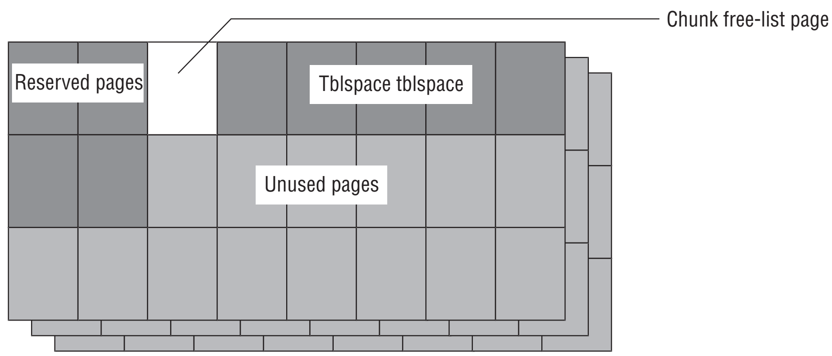


Figure 3-1. Initial Chunk of Regular Dbspace

Structure of an Additional Dbspace Chunk

You can create a dbspace that contains more than one chunk. The initial chunk in a dbspace contains the tblspace **tblspace** for the dbspace. Additional chunks do not. When an additional chunk is first created, it contains the following structures:

- Two reserved pages
- The first chunk free-list page
- Unused pages

Figure 3-2 illustrates the structure of all additional chunks in a dbspace. (The structure also applies to additional chunks in the root dbspace.)

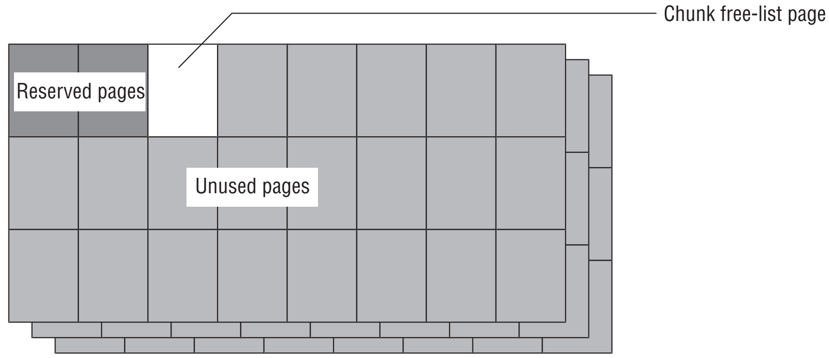


Figure 3-2. Additional Dbspace Chunk

Structure of a Mirror Chunk

Each mirror chunk must be the same size as its primary chunk. When a mirror chunk is created, the database server writes the contents of the primary chunk to the mirror chunk immediately.

The mirror chunk contains the same control structures as the primary chunk. Mirrors of blobspace, sbspace, or dbspace chunks contain the same physical contents as their primary counterpart after the database server brings them online.

Figure 3-3 illustrates the mirror-chunk structure as it appears after the chunk is created.

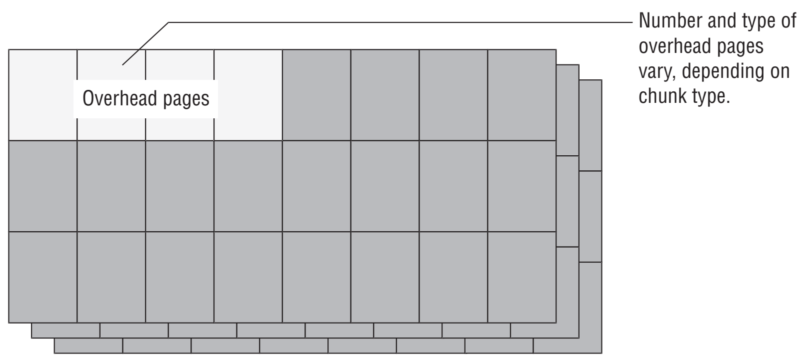


Figure 3-3. Mirror-Chunk Structure

The mirror-chunk structure always shows no free space because all of its space is reserved for mirroring. For more information, see the chapter on what is mirroring in the *IBM Informix Administrator's Guide*.

Structure of the Chunk Free-List Page

In every chunk, the page that follows the last reserved page is the first of one or more chunk free-list pages that tracks available space in the chunk. For a non-root chunk, the initial length of the free space is equal to the size of the chunk minus three pages. If an additional chunk free-list page is needed to accommodate new entries, a new chunk free-list page is created in one of the free pages in the chunk. Figure 3-4 illustrates the location of the free-list page.

Use **oncheck -pe** to obtain the physical layout of pages in the chunk. For more information, see “Check the Chunk Free List with -ce and -pe” on page 6-10.

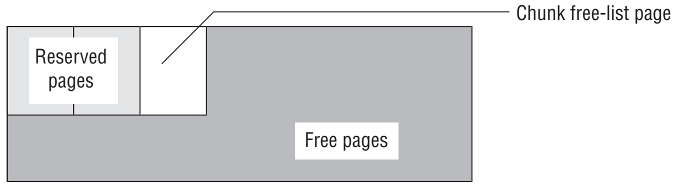


Figure 3-4. Free-List Page

Structure of the Tblspace Tblspace

Each dbspace contains a tblspace called the *tblspace* ***tblspace*** that describes all tblspaces in the dbspace. When the database server creates a tblspace, it places an entry in the tblspace **tblspace** that describes the characteristics of the newly created tblspace. You cannot drop or move a chunk containing a tblspace **tblspace.**

A dbspace can have a maximum number of 2**20 tblspaces.

The default size of the first and next extents depends on whether the dbspace is the root dbspace or not, as shown in the following table.

Table 3-1.

Type of dbspace	Default Size of First Extent	Default Size of Next Extents
Root	• 500 KB for a 2-byte page system • 1000 KB for a 4-byte page system	• 100 KB for a 2-byte page system • 200 KB for a 4-byte page system
Non-root	• 100 KB for a 2-byte page system • 200 KB for a 4-byte page system	• 100 KB for a 2-byte page system • 200 KB for a 4-byte page system

You can specify a non-default size for the first and next extents for a tblspace **tblspace** in the following ways:

- For the root dbspace, set the TBLTBLFIRST and TBLTBLNEXT configuration parameters.
- For non-root dbspaces, use the **onspaces** utility **-ef** and **-en** options when you create a dbspace.

Tblspace Tblspace Entries

To display information on the tblspace, use the **oncheck -pt** command. For more information, see “Display Tblspaces for a Table or Fragment with -pt and -pT” on page 6-15.

Component	Description
Page header	24 bytes, standard page-header information
Page-ending time stamp	4 bytes
Tblspace header	68 bytes, general tblspace information
Column information	Each special column in the table is tracked with an 12-byte entry. (A special column is defined as a VARCHAR, BYTE, or TEXT data type.)
Tblspace name	80 bytes, *database.owner.tablename*

Component	Description
Index information	Each index on the table contains a 20-byte header that contains general information about the index, followed by a 4-byte entry for each column component of the index
Extent information	Each extent allocated to this tblspace is tracked with a 12-byte entry

Tblspace Numbers

Each tblspace that is described in the tblspace tblspace receives a tblspace number. This tblspace number is the same value that is stored as the **partnum** field in the **systables** system catalog table and as the **partn** field in the **sysfragments** system catalog table.

The following SQL query retrieves the **partnum** for every table in the database (these can be located in several different dbspaces) and displays it with the table name and the hexadecimal representation of **partnum**:

```
SELECT tabname, tabid, partnum, HEX(partnum) hex_tblspace_name FROM systables
```

If the output includes a row with a table name but a **partnum** of 0, this table consists of two or more table fragments, each located in its own tblspace. For example, Figure 3-5 shows a table called **account** that has **partnum** 0.

```
tabname        tabid   partnum          hex_tblspace_name

sysfragments     25     1048611          0x00100023
branch          100     1048612          0x00100024
teller          101     1048613          0x00100025
account         102           0          0x00000000
history         103     1048615          0x00100027
results         104     1048616          0x00100028
```

Figure 3-5. Output from systables Query with partnum Values

To obtain the actual tblspace numbers for the fragments that make up the table, you must query the **sysfragments** table for the same database. Figure 3-6 shows that the **account** table from Figure 3-5 has three table fragments and three index fragments.

```
tabid  fragtype     partn          hex_tblspace_name

  102  T            1048614        0x00100026
  102  T            2097154        0x00200002
  102  T            3145730        0x00300002
  102  I            1048617        0x00100029
  102  I            2097155        0x00200003
  102  I            3145731        0x00300003
```

Figure 3-6. Output from sysfragments Table with partn Values

Tblspace Number Elements

The first page in a tblspace is logical page 0. (Physical page numbers refer to the address of the page in the chunk.) The root space tblspace **tblspace** is always contained in the first dbspace and on logical page 1 within the tblspace **tblspace**. (The bitmap page is page 0.)

Tblspace Tblspace Size

These tblspace **tblspace** pages are allocated as an extent when the dbspace is initialized. If the database server attempts to create a table, but the tblspace **tblspace** is full, the database server allocates a next extent to the tblspace.

When a table is removed from the dbspace, its corresponding entry in the tblspace **tblspace** is deleted.

Tblspace Tblspace Bitmap Page

The first page of the tblspace **tblspace**, like the first page of any initial extent, is a bitmap that describes the page fullness of the following pages. Each page that follows has an entry on the bitmap page. If needed, additional bitmap pages are located throughout the contiguous space allocated for the tblspace, arranged so that each bitmap describes only the pages that follow it, until the next bitmap or the end of the dbspace. Bitmap pages fall at distinct intervals within tblspaces pages. Each bitmap page describes a fixed number of pages that follow it.

Structure of the Database Tblspace

The database tblspace appears only in the initial chunk of the root dbspace. The database tblspace contains one entry for each database managed by the database server. Figure 3-7 illustrates the location of the database tblspace.

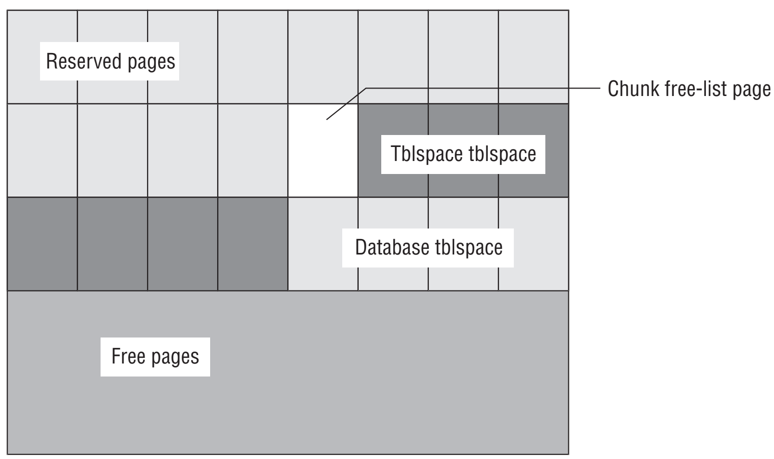


Figure 3-7. Database Tblspace Location in Initial Chunk of Root Dbspace

Database Tblspace Number

The tblspace number of the database tblspace is always 0x100002. This tblspace number appears in an **onstat -t** listing if the database tblspace is active.

Database Tblspace Entries

Each database tblspace entry includes the following five components:

- Database name
- Database owner
- Date and time that the database was created
- The tblspace number of the **systables** system catalog table for this database
- Flags that indicate logging mode

The database tblspace includes a unique index on the database name to ensure that every database is uniquely named. For any database, the **systables** table describes each permanent table in the database. Therefore, the database tblspace only points to the detailed database information located elsewhere.

When the root dbspace is initialized, the database tblspace first extent is allocated. The initial-extent size and the next-extent size for the database tblspace are four pages. You cannot modify these values.

Structure and Allocation of an Extent

This section covers the following topics:

- Extent structure
- Next-extent allocation

Extent Structure

An extent is a collection of contiguous pages within a dbspace. Every permanent database table has two extent sizes associated with it. The initial-extent size is the number of kilobytes allocated to the table when it is first created. The next-extent size is the number of kilobytes allocated to the table when the initial extent, and every extent thereafter, becomes full.

Blobspaces do not use extents.

For specific instructions on how to specify and calculate the size of an extent, see your *IBM Informix Performance Guide*.

Extent Size: The minimum size of an extent is four pages. The default size of an extent is eight pages. The maximum size of an extent is 2**31 pages, equivalent to the maximum chunk size. If the chunk is smaller than the maximum size, the maximum extent size depends on the contiguous space available in the chunk.

Tblspaces that hold *index fragments* follow different rules for extent size. The database server bases the extent size for these tblspaces on the extent size for the corresponding table fragment. The database server uses the ratio of the row size to index key size to assign an appropriate extent size for the index tblspace (see the sections on estimating index page size and fragmenting table indexes in the *IBM Informix Performance Guide*).

Page Types Within a Table Extent: Within the extent, individual pages contain different types of data. Extent pages for a table can be separated into the following categories:

- Data pages

  Data pages contain the data rows for the table.
- Bitmap pages

  Bitmap pages contain control information that monitors the fullness of every page in the extent.
- Blobpages

  Blobpages contain TEXT and BYTE data that is stored with the data rows in the dbspace. TEXT and BYTE data that resides in a blobspace is stored in blobpages, a structure that is completely different than the structure of a dbspace blobpage.
- Free pages

  Free pages are pages in the extent that are allocated for tblspace use, but whose function has not yet been defined. Free pages can be used to store any kind of information: data, including TEXT or BYTE data types; index; or bitmap.

Figure 3-8 on page 3-9 illustrates the possible structure of a nonfragmented table with an initial-extent size of 8 pages and a next-extent size of 16 pages.

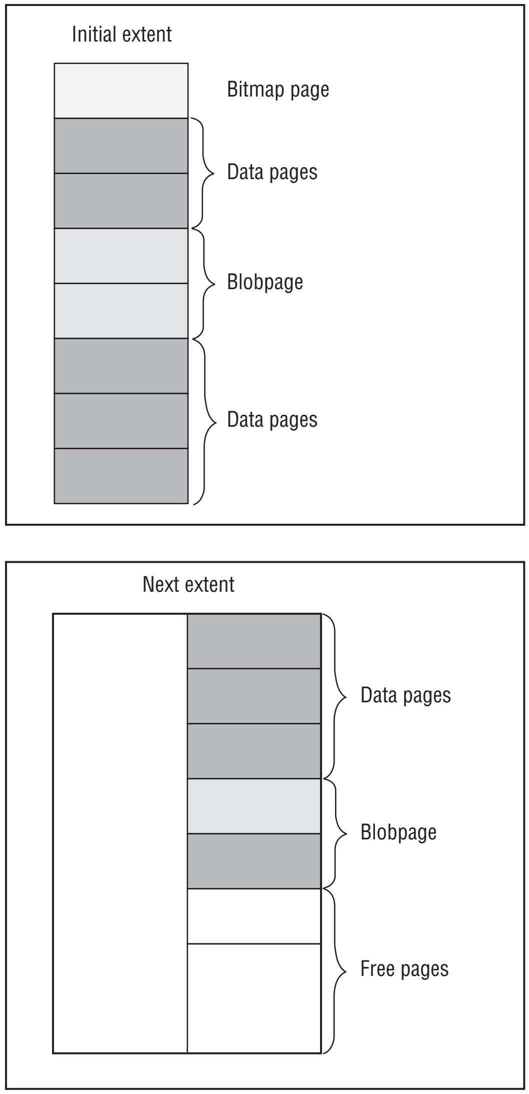


Figure 3-8. Extent Structure of a Table

Page Types Within an Index Extent: The database server stores index pages into different tblspaces than the table with which it is associated. Within the extent, individual index pages contain different types of data. Index pages can be separated into the following categories:

- Index pages (root, branch, and leaf pages)

  Index pages contain the index information for the table.
- Bitmap pages

  Bitmap pages contain control information that monitors the fullness of every page in the extent.
- Free pages

  Free pages are pages in the extent that are allocated for tblspace use, but whose function has not yet been defined. Free pages can be used to store any kind of information: data, index, TEXT or BYTE data, or bitmap.

All indexes are detached unless you explicitly specify attached indexes.

Important: An extent that is allocated for a table fragment does not contain index pages. Index pages for a fragmented table always reside in a separate tblspace. For more information, see fragmenting table indexes in the chapter on table fragmentation and PDQ in the *IBM Informix Administrator's Guide*.

Figure 3-9 on page 3-10 illustrates the extent structure of an index.

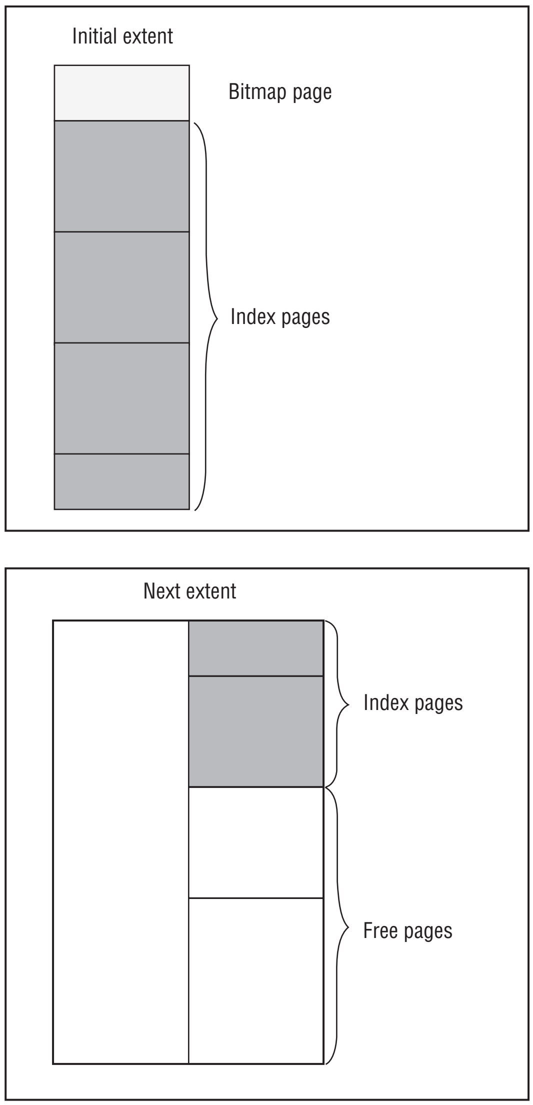


Figure 3-9. Extent Structure of an Index

Next-Extent Allocation

After the initial extent fills, the database server attempts to allocate another extent of contiguous disk space. The procedure that the database server follows is referred to as next-extent allocation.

Extents for a tblspace are tracked as one component of the tblspace **tblspace** information for the table. The maximum number of extents allocated for any tblspace is application and machine dependent because it varies with the amount of space available on the tblspace **tblspace** entry.

Next-Extent Size: The number of kilobytes that the database server allocates for a next extent is, in general, equal to the size of a next extent, as specified in the SQL statement CREATE TABLE. However, the actual size of the next-extent allocation might deviate from the specified size because the allocation procedure takes into account the following three factors:

- Number of existing extents for this tblspace
- Availability of contiguous space in the chunk and dbspace
- Location of existing tblspace extents

The effect of each of these factors on next-extent allocation is explained in the paragraphs that follow and in Figure 3-10 on page 3-12.

Extent Size Doubling: If a permanent table or user-defined temporary table already has 16 extents allocated, the database server automatically doubles the size for subsequent allocations. This doubling occurs every 16 extents. For example, if you create a table with NEXT SIZE equal to 20 kilobytes, the database server allocates the first 16 extents at a size of 20 kilobytes each. The database server allocates extents 17 to 32 at 40 kilobytes each, extents 33 to 48 at 80 kilobytes each, and so on.

The extent size doubling is allowed only if the total number of pages allocated so far is at least 16 times the current next extent size. This is a precautionary measure to limit the exponential doubling of next extent size, if the system has many small holes of pages less than the next extent size, thereby creating a greater number of small extents.

For system-created temporary tables, the next-extent size begins to double after 4 extents have been added.

Lack of Contiguous Space: If the database server cannot find available contiguous space in the first chunk equal to the size specified for the next extent, it extends the search to the next chunk in the dbspace. Extents are not allowed to span chunks.

If the database server cannot find adequate contiguous space anywhere in the dbspace, it allocates to the table the largest available amount of contiguous space. (The minimum allocation is four pages. The default value is eight pages.) No error message is returned if an allocation is possible, even when the amount of space allocated is less than the requested amount.

Merge of Extents for the Same Table: If the disk space allocated for a next extent is physically contiguous with disk space already allocated to the same table, the database server allocates the disk space but does not consider the new allocation as a separate extent. Instead, the database server extends the size of the existing contiguous extent. Thereafter, all disk-space reports reflect the allocation as an extension of the existing extent. That is, the number of extents reported is always the number of physically distinct extents, not the number of times a next extent has been allocated plus one (the initial extent). Figure 3-10 illustrates extent-allocation strategies.

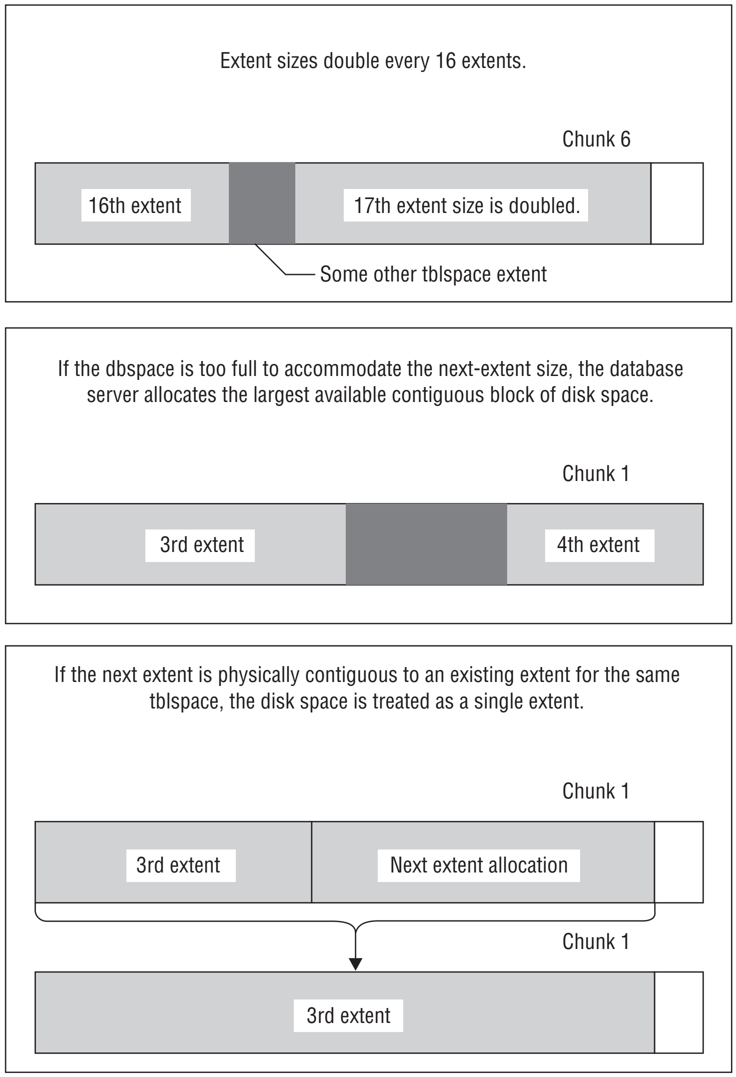


Figure 3-10. Next-Extent Allocation Strategies

After disk space is allocated to a tblspace as part of an extent, the space remains dedicated to that tblspace even if the data contained in it is deleted. For alternative methods of reclaiming this empty disk space, see your *IBM Informix Performance Guide*.

Structure and Storage of a Dbspace Page

The basic unit of database server I/O is a page. Page size might vary among computers.

In Dynamic Server, the page size depends on the operating system.

Rows in Nonfragmented Tables

The database server can store rows that are longer than a page. The database server also supports the VARCHAR data type, which results in rows of varying length. As a result, rows do not conform to a single format.

Rows within a table are not necessarily the same length if the table contains one or more columns of type VARCHAR. In addition, the length of a row in such a table might change when an end user modifies data contained in the VARCHAR column.

The length of a row can be greater than a page.

TEXT and BYTE data is not stored within the data row. Instead, the data row contains a 56-byte descriptor that points to the location of the data. The descriptor can point to a dbspace page.

The descriptor can point to a blobspace blobpage. If you are using the Optical Subsystem, the descriptor can also point to an optical-storage subsystem.

For instructions about how to estimate the length of fixed-length and variable-length data rows, see your *IBM Informix Performance Guide*.

Definition of Rowid: Informix uses two different types of rowids to identify data in tables:

- *Serial rowid*

  These rowids are fields in a table and are assigned to tables created with the WITH ROWID option.

- *Internal rowid*

  The database server identifies each data row in a table with a unique internal rowid. This rowid identifies the location of the row within the dbspace.

  To obtain the internal rowids for a table, use the **oncheck -pD** option. For more information, see “Check Pages with -cd and -cD” on page 6-8.

In a nonfragmented table, the term *rowid* refers to a unique 4-byte integer that defines the physical location of the row in the table. The page that contains the first byte of the data row is the page that is specified by the rowid. This page is called the data row *home page*.

Fragmented tables can also have rowids, but they are implemented in a different way. For more information on this topic, see “Rows in Fragmented Tables” on page 3-13.

Use of Rowids: Every data row in a nonfragmented table is uniquely identified by an unchanging rowid. When you create an index for a nonfragmented table, the rowid is stored in the index pages associated with the table to which the data row belongs. When the database server requires a data row, it searches the index to find the key value and uses the corresponding rowid to locate the requested row. If the table is not indexed, the database server might sequentially read all the rows in the table.

Eventually, a row might outgrow its original storage location. If this occurs, a *forward pointer* to the new location of the data row is left at the position defined by the rowid. The forward pointer is itself a rowid that defines the page and the location on the page where the data row is now stored.

Rows in Fragmented Tables

Unlike rows in a nonfragmented table, the database server does *not* assign a rowid to rows in fragmented tables. If you want to access data by rowid, you must explicitly create a rowid column as described in your *IBM Informix Performance*

Guide. If user applications attempt to reference a rowid in a fragmented table that does not contain a rowid that you explicitly created, the database server returns an appropriate error code to the application.

Access to Data in Fragmented Tables with Rowid: From the viewpoint of an application, the functionality of a rowid column in a fragmented table is identical to the rowid of a nonfragmented table. However, unlike the rowid of a nonfragmented table, the database server uses an index to map the rowid to a physical location.

When the database server accesses a row in a fragmented table using the rowid column, it uses this index to look up the physical address of the row before it attempts to access the row. For a nonfragmented table, the database server uses direct physical access without an index lookup. As a consequence, accessing a row in a fragmented table using rowid takes slightly longer than accessing a row using rowid in a nonfragmented table. You should also expect a small performance impact on the processing of inserts and deletes due to the cost of maintaining the rowid index for fragmented tables.

Primary-key access can lead to significantly improved performance in many situations, particularly when access is in parallel.

Recommendations on Use of Rowid

It is recommended that application developers use primary keys as a method of access rather than rowids. Because primary keys are defined in the ANSI specification of SQL, using them to access data makes your applications more portable.

For a complete description on how to define and use primary keys to access data, see the *IBM Informix Guide to SQL: Reference* and the *IBM Informix Guide to SQL: Tutorial*.

Data-Row Format and Storage

The variable length of a data row has the following consequences for row storage:

- A page might contain one or more whole rows.
- A page might contain portions of one or more rows.
- A page might contain a combination of whole rows and partial rows.
- An updated row might increase in size and become too long to return to its original storage location in a row.

The following paragraphs describe the guidelines that the database server follows during data storage.

Storage of Row: To minimize retrieval time, rows are not broken across page boundaries unnecessarily. Rows that are shorter than a page are always stored as whole rows. A page is considered *full* when the count of free bytes is less than the number of bytes needed to store a row of maximum size.

Location of Rows: When the database server receives a row that is longer than a page, the row is stored in as many whole pages as required. The database server then stores the trailing portion in less than a full page.

The page that contains the first byte of the row is the row home page. The number of the home page becomes the logical page number contained in the rowid. Each full page that follows the home page is referred to as a big-remainder page. If the trailing portion of the row is less than a full page, it is stored on a remainder page.

After the database server creates a remainder page to accommodate a long row, it can use the remaining space in this page to store other rows.

Figure 3-11 illustrates the concepts of home page, big-remainder page, and remainder page.

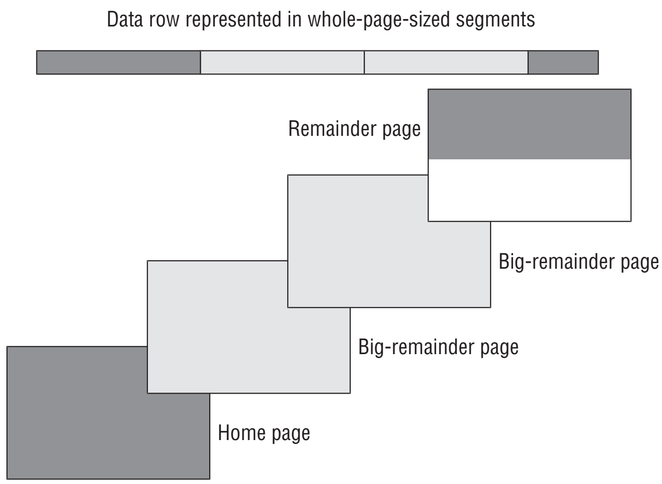


Figure 3-11. Remainder Pages

Page Compression: Over time, the free space on a page can become fragmented. When the database server attempts to store data, it first checks row length against the number of free bytes on a page to determine if the row fits. If adequate space is available, the database server checks if the page contains adequate contiguous free space to hold the row (or row portion). If the free space is not contiguous, the database server calls for page compression.

Structure of Fragmented Tables

Although table fragmentation is transparent to applications, as database server administrator you should be aware of how the database server allocates disk space for table fragments and how the database server identifies rows in those fragments.

Each table fragment has its own tblspace with a unique *tblspace_id* or *fragment_id*. Figure 3-12 shows the disk allocation for a fragmented table that resides in different partitions of the same dbspace.

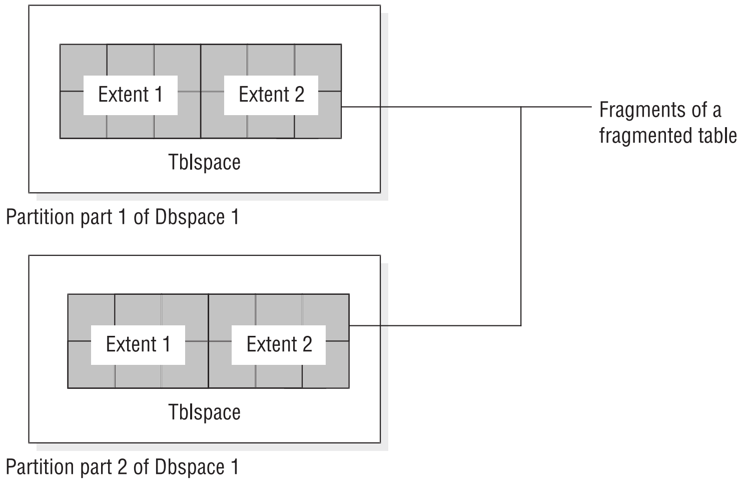


Figure 3-12. Disk Structures for a Fragmented Table

Attached Indexes

With an attached index, the index and data are fragmented in the same way. You can decide whether to store the index pages with the corresponding data pages in the same dbspace or store them in separate dbspaces. For information on choosing a fragmentation strategy, see the *IBM Informix Performance Guide*.

Detached Indexes

For detached indexes, the table fragment and index fragment are stored in tblspaces in separate dbspaces.

Structure of B-Tree Index Pages

This section provides general information about the structure of B-tree index pages. It is designed as an overview for the interested reader. For more information on B-tree indexes, see your *IBM Informix Performance Guide*.

Definition of B-Tree Terms

The database server uses a B-tree structure to organize index information. Figure 3-13 shows that a fully developed B-tree index is composed of the following three different types of index pages or nodes:

- One *root node*

  A root node contains node pointers to branch nodes.
- Two or more *branch nodes*

  A branch node contains pointers to leaf nodes or other branch nodes.
- Many *leaf nodes*

  A leaf node contains index items and horizontal pointers to other leaf nodes.

Each node serves a different function. The following sections describe each node and the role that it plays in indexing.

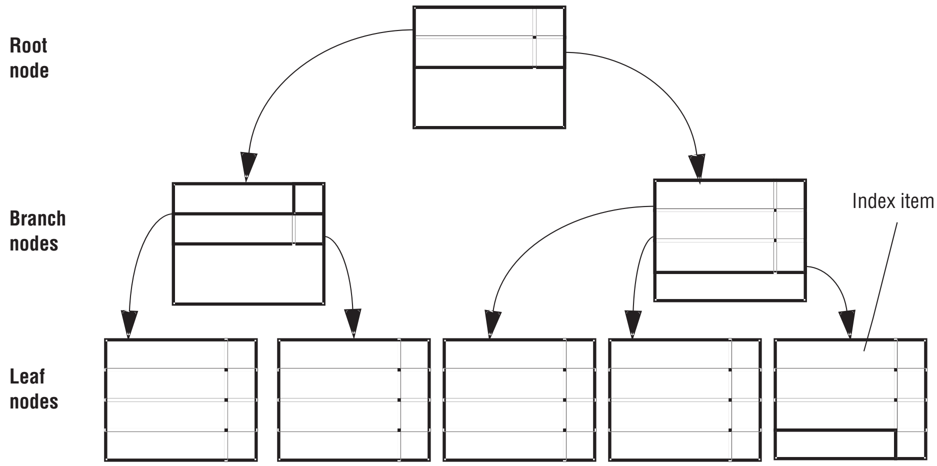


Figure 3-13. Full B-Tree Structure

Index Items: The fundamental unit of an index is the *index item.* An index item contains a key value that represents the value of the indexed column for a particular row. An index item also contains rowid information that the database server uses to locate the row in a data page.

Nodes: A node is an index page that stores a group of index items. For the three types of nodes, see “Definition of B-Tree Terms” on page 3-16.

Logical Storage of Indexes

This section presents an overview of how the database server creates and fills an index.

Creation of Root and Leaf Nodes: When you create an index for an empty table, the database server allocates a single index page. This page represents the root node and remains empty until you insert data in the table.

At first, the root node functions in the same way as a leaf node. For each row that you insert into the table, the database server creates and inserts an index item in the root node. Figure 3-14 illustrates how a root node appears before it fills.

Root node 1

Albertson	rowid information
Baxter	rowid information
Beatty	rowid information
Currie	rowid information
Keyes	rowid information
Lawson	rowid information
Mueller	rowid information
Wallach	rowid information

Figure 3-14. Root Node

When the root node becomes full of index items, the database server splits the root node by performing the following steps:

- Creates two leaf nodes
- Moves approximately half of the root-node entries to each of the newly created leaf nodes
- Puts pointers to leaf nodes in the root node

As you add new rows to a table, the database server adds index items to the leaf nodes. When a leaf node fills, the database server creates a new leaf node, moves part of the contents of the full index node to the new node, and adds a node pointer to the new leaf node in the root node.

For example, suppose that leaf node 3 in Figure 3-15 becomes full. When this situation occurs, the database server adds yet another leaf node. The database server moves part of the records from leaf node 3 to the new leaf node, as Figure 3-15 shows.

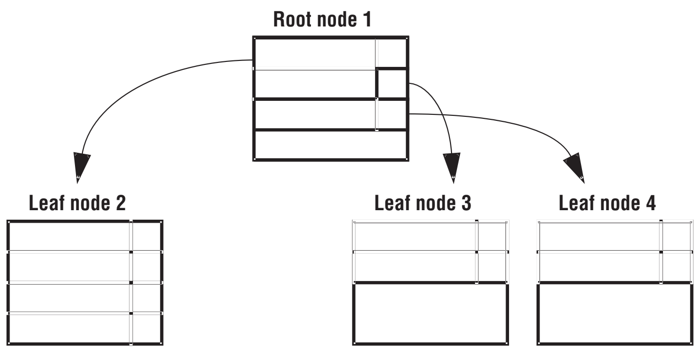


Figure 3-15. Leaf Node 4 Created After Leaf Node 3 Fills

Creation of Branch Nodes: Eventually, as you add rows to the table, the database server fills the root node with node pointers to all the existing leaf nodes. When the database server splits yet another leaf node, and the root node has no room for an additional node pointer, the following process occurs.

The database server splits the root node and divides its contents among two newly created branch nodes. As index items are added, more and more leaf nodes are split, causing the database server to add more branch nodes. Eventually, the root node fills with pointers to these branch nodes. When this situation occurs, the database server splits the root node again. The database server then creates yet another branch level between the root node and the lower branch level. This process results in a four-level tree, with one root node, two branch levels, and one leaf level. The B-tree structure can continue to grow in this way to a maximum of 20 levels.

Branch nodes can point either to other branch nodes below them (for large indexes of four levels or more) or to leaf nodes. In Figure 3-16, the branch node points to leaf nodes only. The first item in the left branch node contains the same key value as the largest item in the leftmost leaf node and a node pointer to it. The second item contains the largest item in the next leaf node and a node pointer to it. The third item in the branch node contains only a pointer to the next higher leaf node. Depending on the index growth, this third item can contain the actual key value in addition to the pointer at a later point during the lifespan of the index.

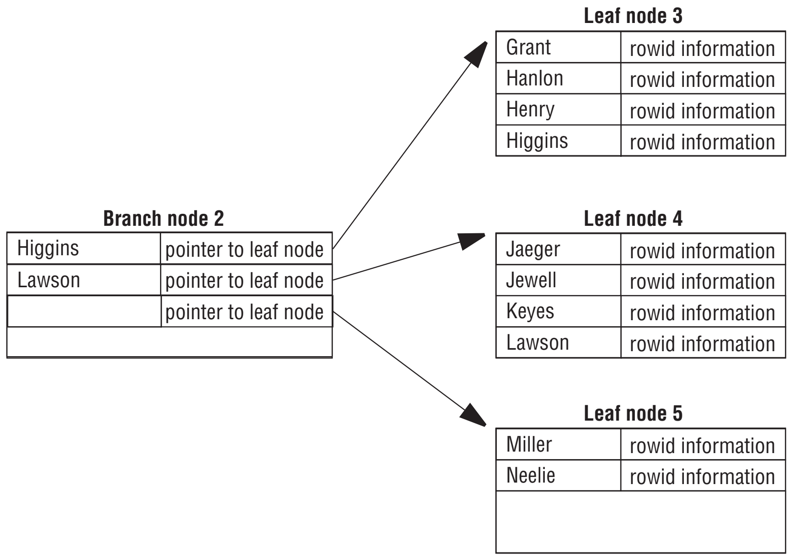


Figure 3-16. Typical Contents of a Branch Node

Duplicate Key Values: Duplicate key values occur when the value of an indexed column is identical for multiple rows. For example, suppose that the third and fourth leaf nodes of a B-tree structure contain the key value `Smith`. Suppose further that this value is duplicated six times, as Figure 3-17 illustrates.

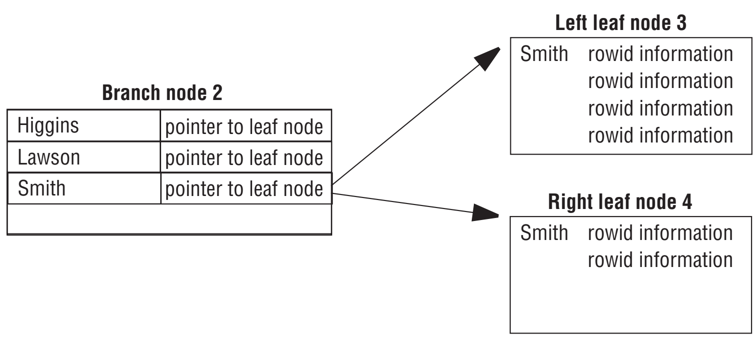


Figure 3-17. Leaf Nodes 3 and 4

The first item on the third leaf page contains the duplicate key value, `Smith`, and the rowid information for the first physical row in the table that contains the duplicate key value. To conserve space, the second item does not repeat the key value `Smith` but instead contains just the rowid information. This process continues throughout the page; no other key values are on the leaf, only rowid information.

The first item on the fourth leaf page again contains the duplicated key value and rowid information. Subsequent items contain only rowid information.

Now consider the branch node. The third item in the branch node contains the same key value and rowid as the largest item in the third leaf node and a node pointer to it. The fourth item would contain only a node pointer to the fourth leaf node, thus saving the space of an additional duplicate key value.

Key-Value Locking: To increase concurrency, the database server supports *key-value* locking in the B-tree index. Key-value locking locks only the value of the key instead of the physical location in the B-tree index.

One of the most important uses for key-value locking is to assure that a unique key remains unique through the end of the transaction that deleted it. Without this protection mechanism, user A might delete a unique key within a transaction, and user B might insert a row with the same key before the transaction commits. This scenario makes rollback by user A impossible. Key-value locking prevents user B from inserting the row until the end of user A's transaction.

Adjacent Key Locking: With Repeatable Read isolation level, the database server is required to protect the *read set*. The read set consists of the rows that meet the filters in the WHERE clause of the query. To guarantee that the rows do not change, the database server obtains a lock on the index item that is adjacent to the right-most item of the read set.

Freed Index Pages: When the database server physically removes an index item from a node and frees an index page, the freed page is reused.

Filling Indexes: When you create an index, you can specify how densely or sparsely filled you want the index. The index fill factor is a percentage of each index page that will be filled during the index build. Use the FILLFACTOR option of the CREATE INDEX statement or the FILLFACTOR configuration parameter to set the fill factor. This option is particularly useful for indexes that you do not expect to grow after they are built. For additional information about the FILLFACTOR option of the CREATE INDEX statement, see the *IBM Informix Guide to SQL: Syntax*.

Calculating the Length of Index Items: For data types other than VARCHAR, the length of an index item is calculated by adding the length of the key value plus 5 bytes for each rowid information associated with the key value.

The key values in an index are typically of fixed length. If an index holds the value of one or more columns of the VARCHAR data type, the length of the key value is at least the sum of the length-plus-one of each VARCHAR value in the key.

In Dynamic Server, the maximum length of a key value is 390 bytes. The combined size of VARCHAR columns that make up a key must be less than 390, minus an additional byte for each VARCHAR column. For example, the key length of the index that the database server builds for the following statements equals 390, or `((255+1) + (133+1))`:

```
CREATE TABLE T1 (c1 varchar(255, 10), c2 varchar(133, 10));
CREATE INDEX I1 on T1(c1, c2);
```

Functional Indexes

A *functional index* is one in which all keys derive from the results of a function. If you have a column of pictures, for example, and a function to identify the predominant color, you can create an index on the result of the function. Such an index would enable you to quickly retrieve all pictures having the same predominant color, without re-executing the function.

A functional index uses the same B-tree structure as any other B-tree index. The only difference is that the determining function is applied during an insert or an update whenever the column that is the argument to the function changes. For more information on the nature of functional indexes, refer to your *IBM Informix Performance Guide*.

To create a functional index, use the CREATE FUNCTION and CREATE INDEX statements. For more information on these statements, refer to the *IBM Informix Guide to SQL: Syntax*.

Structure of R-Tree Index Pages

An index structure that relies on one-dimensional ordering of key values does not work for spatial data; for example, two dimensional geometric shapes such as circles, squares, and triangles. Efficient retrieval of spatial data, such as the data used in geographic information systems (GIS) and computer-aided design (CAD) applications, requires an access method that handles multidimensional data. The database server implements an R-tree index to access spatial data efficiently. For information about the structure of index pages, refer to the *IBM Informix R-Tree Index User's Guide*.

Storage of Simple Large Objects

This section explains the structures and storage techniques that the database server uses to store simple large objects (TEXT or BYTE data).

Structure of a Blobspace

When you create a blobspace, you can specify the effective size of the data pages, which are called blobpages. The blobpage size for the blobspace is specified when the blobspace is created. Blobpage size must be a multiple of page size. (For information on determining database server page size, see the chapter on managing disk space in the *IBM Informix Administrator's Guide*.) All blobpages within a blobspace are the same size, but the size of the blobpage can vary between blobspaces. Blobpage size can be greater than the page size because data stored in a blobspace is never written to the page-sized buffers in shared memory.

The advantage of customizing the blobpage size is storage efficiency. Within a blobspace, TEXT and BYTE data is stored in one or more blobpages, but simple large objects do not share blobpages. Storage is most efficient when the TEXT or BYTE data is equal to or slightly smaller than the blobpage size.

The blobspace free-map pages and bitmap pages are the size specified as a database server page, which enables them to be read into shared memory and to be logged.

When the blobspace is first created, it contains the following structures:

- Blobspace free-map pages
- The blobspace bitmap that tracks the free-map pages
- Unused blobpages

Structure of a Dbspace Blobpage

TEXT or BYTE data that is stored in the dbspace is stored in a blobpage. The structure of a dbspace blobpage is similar to the structure of a dbspace data page. The only difference is an extra 12 bytes that can be stored along with the TEXT or BYTE data in the data area.

Simple large objects can share dbspace blobpages if more than one simple large object can fit on a single page, or if more than one trailing portion of a simple large object can fit on a single page.

For a discussion of how to estimate the number of dbspace blobpages needed for a specific table, see your *IBM Informix Performance Guide*.

Each segment of TEXT or BYTE data stored in a dbspace page might be preceded by up to 12 bytes of information that does not appear on any other dbspace page. These extra bytes are overhead.

Simple-Large-Object Storage and the Descriptor

Data rows that include TEXT or BYTE data do not include the data in the row itself. Instead, the data row contains a 56-byte descriptor with a forward pointer (rowid) to the location where the first segment of data is stored.

The descriptor can point to one of the following items:
- A page (if the data is stored in a dbspace)
- A blobpage (if the data is stored in a blobspace)
- An optical platter (if you are using the Optical Subsystem)

Creation of Simple Large Objects

When a row that contains TEXT or BYTE data is to be inserted, the simple large objects are created first. After the simple large objects are written to disk (or optical medium), the row is updated with the descriptor and inserted.

Deletion or Insertion of Simple Large Objects

The database server cannot modify simple large objects. It can only insert or delete them. Deleting a simple large object means that the database server frees the space consumed by the deleted object for reuse.

When TEXT or BYTE data is updated, a new simple large object is created, and the data row is updated with the new blob descriptor. The old image of the row contains the descriptor that points to the obsolete value for the simple large object. The space consumed by the obsolete simple large object is freed for reuse after the update is committed. Simple large objects are automatically deleted if the rows that contain their blob descriptors are deleted. (Blobpages that stored a deleted simple large object are not available for reuse until the logical log that contains the original INSERT record for the deleted simple large object is backed up. For more information, see backing up logical-log files to free blobpages in the chapter on what is the logical log in the *IBM Informix Administrator's Guide*.)

Size Limits for Simple Large Objects

The largest simple large object that the blob descriptor can accommodate is (2^{31} - 1), or about 2 gigabytes.

Blobspace Page Types

Every blobspace chunk contains three types of pages:
- A blobspace free-map page
- A bitmap page
- Blobpages

Blobspace Free-Map Page

The blobspace free-map page identifies unused blobpages so that the database server can allocate them as part of simple-large-object creation. When a blobpage is allocated, the free-map entry for that page is updated. All entries for a single simple large object are linked.

A blobspace free-map page is the size of one database server page. Each entry on a free-map page is 8 bytes, stored as two 32-bit words, as follows:

- The first bit in the first word specifies whether the blobpage is free or used.
- The next 31 bits in the first word identify the logical-log file that was current when this blobpage was written. (This information is needed for logging TEXT or BYTE data.)
- The second word contains the tblspace number associated with the simple large object stored on this page.

The number of entries that can fit on a free-map page depends on the page size of your computer. The number of free-map pages in a blobspace chunk depends on the number of blobpages in the chunk.

Blobspace Bitmap Page

The blobspace bitmap page tracks the fullness and number of blobspace free-map pages in the chunk. Each blobspace bitmap page is capable of tracking a quantity of free-map pages that represent more than 4,000,000 blobpages. Each blobspace bitmap page is the size of one page.

Blobpage

The blobpage contains the TEXT or BYTE data. Blobpage size is specified by the database server administrator who creates the blobspace. Blobpage size is specified as a multiple of the page size.

Structure of a Blobspace Blobpage

The storage strategy used to store simple large objects in a blobspace differs from the dbspace storage strategy. The database server does not combine whole simple large objects or portions of a simple large object on a single blobspace blobpage. For example, if blobspace blobpages are 24 kilobytes each, a simple large object that is 26 kilobytes is stored on two 24-kilobyte pages. The extra 22 kilobytes of space remains unused.

The structure of a blobpage includes a blobpage header, the TEXT or BYTE data, and a page-ending time stamp. The blobpage header includes, among other information, the page-header time stamp and the blob time stamp associated with the forward pointer in the data row. If a simple large object is stored on more than one blobpage, a forward pointer to the next blobpage and another blob time stamp are also included in the blobpage header.

Sbspace Structure

An sbspace is similar to a blobspace except that it holds smart large objects.

When an sbspace is created in a database, it contains an sbspace descriptor. Each sbspace chunk contains the following structures:

- Sbspace chunk descriptors
- Chunk free-page list
- An sbspace metadata area (up to one for each chunk)
- Reserved data areas (up to two for each chunk)
- User-data areas (up to two for each chunk)

For best performance, it is recommended that the metadata area be located in the middle of the sbspace. The database server automatically places the metadata area in the correct location. However, to specify the location of the metadata area, specify the **-Mo** flag in the **onspaces** command.

If you do not specify the size of the metadata area in the **-Ms** flag of the **onspaces** command, the database server uses the value of AVG_LO_SIZE (defaults to 8 kilobytes) to calculate the size of the metadata area. For more information, see “Creating an Sbspace with the -Df option” on page 13-11.

Normally, you can let the system calculate the metadata size for you. If you want to estimate the size of the metadata area, see the chapter on table performance considerations in the *IBM Informix Performance Guide*.

Figure 3-18 illustrates the chunk structure of an sbspace as it appears immediately after the sbspace is created. Each reserved area can be allocated to either the user-data or metadata area. Reserved areas are always within the user-data area of the chunk.

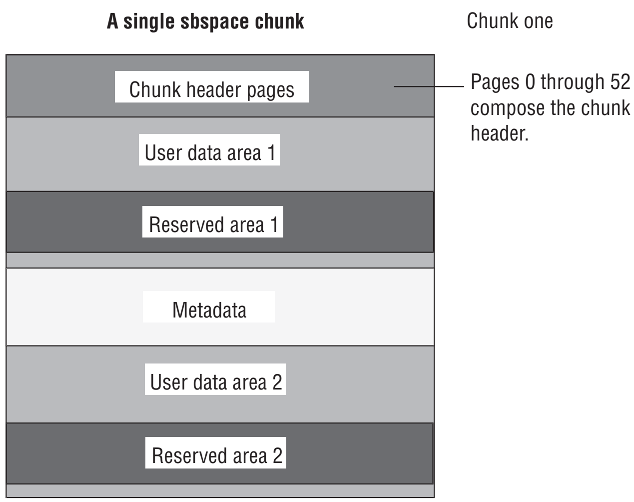


Figure 3-18. A Single Sbspace Chunk

Because the chunk in Figure 3-18 is the first in the sbspace, it contains an sbspace descriptor. The chunk descriptor tblspace in **chunk one** contains information about chunk one and all chunks added to the sbspace thereafter.

Structure of the Metadata Area

As with the chunk header pages, four areas are exclusive to the first chunk in a sbspace: the sbspace descriptor tblspace, the chunk adjunct tblspace, and the level-1 and level-2 archive tblspaces. The tblspace header section contains a tblspace header for each of these tblspaces (notably excluding the tblspace **tblspace**). Figure 3-19 shows the layout of the metadata in the single-chunk sbspace.

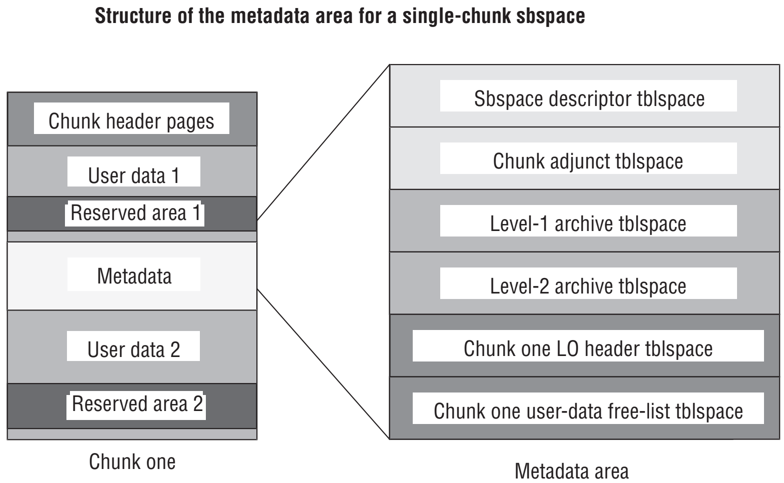


Figure 3-19. Structure of the Metadata Area for a Single-Chunk Sbspace

When you specify the sbspace name in the **oncheck -ps** option, you can display the number of pages allocated and used for each tblspace in the metadata area.

The following describes how the metadata area grows:

- The sbspace descriptor tblspace does not grow.
- The chunk adjunct tblspace grows as chunks are added.
- The LO header tblspace grows as chunks are added.
- The tblspace for user-data free list grows if free spaces in the chunk are heavily fragmented.

Sbpage Structure

Each sbpage is composed of three elements: an sbpage header, the actual user data itself, and an sbpage trailer. Figure 3-20 shows the structure of an sbpage. The sbpage header consists of the standard page header. The sbpage trailer is used to detect an incomplete write on the page and to detect page corruption.

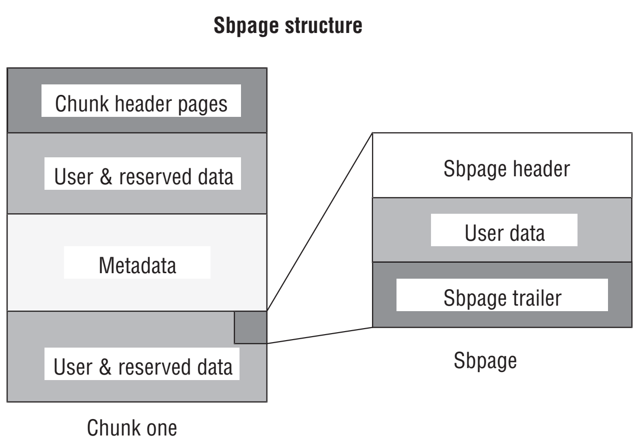


Figure 3-20. Sbpage Structure

Multiple Chunk Sbspace

Figure 3-21 illustrates a possible configuration for a three-chunk sbspace. In this example, **chunk two** contains no metadata of its own. Metadata information for **chunk two** is stored in the metadata area of **chunk one**.

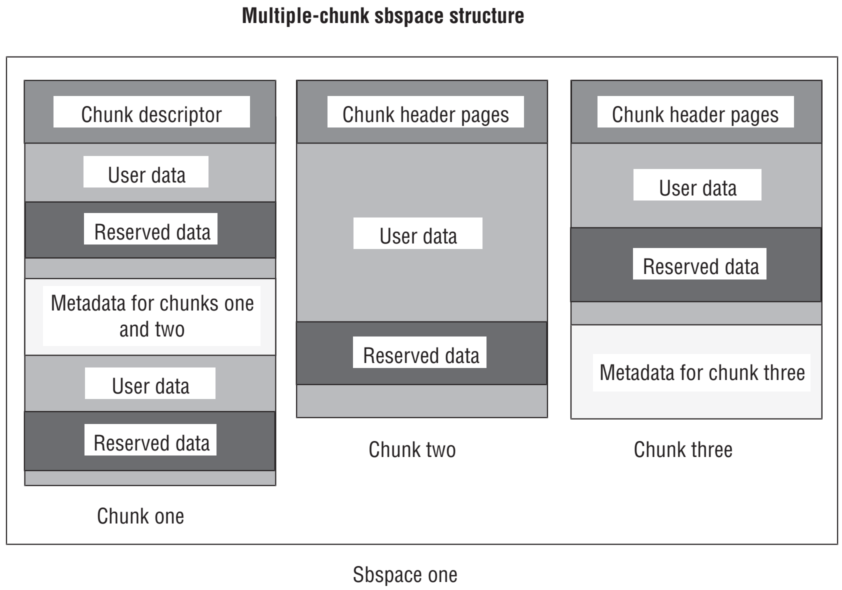


Figure 3-21. Multiple-Chunk Sbspace Structure

The user-data area in **chunk one** of the example is actually optional. **Chunk one** could contain metadata for all other chunks in the sbspace.

Time Stamps

The database server uses a time stamp to identify a time when an event occurred relative to other events of the same kind. The time stamp is not a literal time that refers to a specific hour, minute, or second. It is a 4-byte integer that the database server assigns sequentially.

Database and Table Creation: What Happens on Disk

This section explains how the database server stores data related to the creation of a database or table and allocates the disk structures that are necessary to store your data.

Database Creation

After the root dbspace exists, users can create a database. The paragraphs that follow describe the major events that occur on disk when the database server adds a new database.

Disk-Space Allocation for System Catalog Tables

The database server searches the chunk free-list pages in the dbspace, looking for free space in which to create the system catalog tables. For each system catalog table, in turn, the database server allocates eight contiguous pages, the size of the initial extent of each system catalog table. The tables are created individually and do not necessarily reside next to each other in the dbspace. They can be located in different chunks. As adequate space is found for the initial extent of each table, the pages are allocated, and the associated chunk free-list page is updated.

Tracking of System Catalog Tables

The database server tracks newly created databases in the database tblspace, which resides in the root dbspace. An entry describing the database is added to the database tblspace in the root dbspace. (See "Structure of the Database Tblspace" on page 3-7.) For each system catalog table, the database server adds a one-page entry to the tblspace **tblspace** in the dbspace where the database was built. (See "Structure of the Tblspace Tblspace" on page 3-5.) Figure 3-22 illustrates the relationship between the database tblspace entry and the location of the **systables** system catalog table for the database.

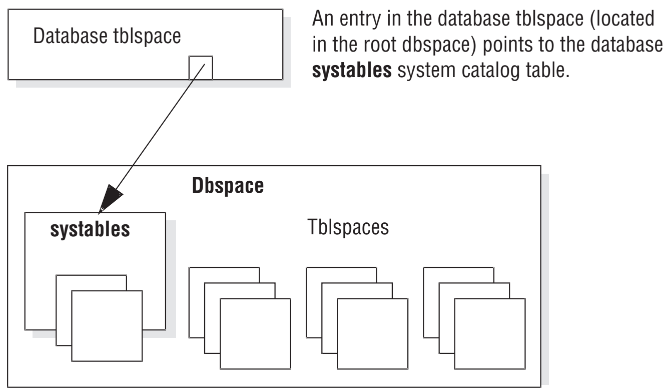


Figure 3-22. New Databases

For instructions on how to list your databases after you create them, see monitoring databases in the chapter on managing database-logging status in the *IBM Informix Administrator's Guide*.

Table Creation

After the root dbspace exists, and a database has been created, users with the necessary SQL privileges can create a database table. When users create a table, the database server allocates disk space for the table in units called extents (see what is an extent in the chapter on where data is stored in the *IBM Informix Administrator's Guide*). The paragraphs that follow describe the major events that occur when the database server creates a table and allocates the initial extent of disk space.

Disk-Space Allocation

The database server searches the chunk free-list pages in the dbspace for contiguous free space equal to the initial extent size for the table. When adequate space is found, the pages are allocated, and the associated chunk free-list page is updated.

If the database server cannot find adequate contiguous space anywhere in the dbspace, it allocates to the table the largest available amount of contiguous space. No error message is returned if an allocation is possible, even when the amount of space allocated is less than the requested amount. If the minimum extent size cannot be allocated, an error is returned. (Extents cannot span two chunks.)

Entry in the Tblspace Tblspace

The database server adds a one-page entry for this table to the tblspace **tblspace** in this dbspace. The tblspace number assigned to this table is derived from the logical page number in the tblspace **tblspace** where the table is described. See “Tblspace Numbers” on page 3-6.

The tblspace number indicates the dbspace where the tblspace is located. Tblspace extents can be located in any of the dbspace chunks.

If you must know exactly where the tblspace extents are located, execute the **oncheck -pe** command for a listing of the dbspace layout by chunk.

Entries in the System Catalog Tables

The table itself is fully described in entries stored in the system catalog tables for the database. Each table is assigned a table identification number or *tabid*. The tabid value of the first user-defined table object in a database is always `100`. (The object whose tabid = 100 might also be a view, synonym, or a sequence.) For a complete discussion of the system catalog, see the *IBM Informix Guide to SQL: Reference*.

A table can be located in a dbspace that is different than the dbspace that contains the database. The tblspace itself is the sum of allocated extents, not a single, contiguous allocation of space. The database server tracks tblspaces independently of the database.

Creation of a Temporary Table

The tasks involved in creating temporary tables are similar to the tasks that the database server performs when it adds a new permanent table. The key difference is that temporary tables do not receive an entry in the system catalog for the database. For more information, see the section defining a temporary table, in the chapter on where data is stored in the *IBM Informix Administrator's Guide*.

Chapter 4. Interpreting Logical-Log Records

In This Chapter

To display the logical-log records that the logical-log files contain, use the **onlog** utility.

This chapter provides the following information:
- Brief guidance on reading logical-log records
- A listing of the different logical-log record types

In general, you do not need to read and interpret your logical-log files. However, **onlog** output is useful in debugging situations. For example, you might want to use **onlog** to track a specific transaction or to see what changes the database server made to a specific tblspace. You can also use **onlog** to investigate the cause of an error that occurs during a rollforward. For more information, see “onlog: Display Logical-Log Contents” on page 9-1.

About Logical-Log Records

Most SQL statements generate multiple logical-log records. Interpreting logical-log records is more complicated when the database server records the following events in the logical log:
- A transaction that drops a table or index
- A transaction that rolls back
- A checkpoint in which transactions are still active
- A distributed transaction

The following sections discuss the logical-log records for these events.

Transactions That Drop a Table or Index

Once the database server drops a table or index from a database, it cannot roll back that drop operation. If a transaction contains a DROP TABLE or DROP INDEX statement, the database server handles this transaction as follows:
1. The database server completes all the other parts of the transaction and writes the relevant logical-log records.
2. The database server writes a BEGCOM record to the logical log and the records associated with the DROP TABLE or DROP INDEX (DINDEX, for example).
3. The database server writes a COMMIT record.

If the transaction is terminated unexpectedly after the database server writes the BEGCOM record to the logical log, the database server rolls *forward* this transaction during recovery because it cannot roll back the drop operation.

Transactions That Are Rolled Back

When a rollback occurs, the database server generates a compensation-log record (CLR) for each record in the logical log that is rolled back. The database server uses the CLRs if a system failure takes place *during a rollback*. The CLRs provide the database server with information on how far the rollback progressed before the failure occurred. In other words, the database server uses the CLRs to log the rollback.

If a CLR contains the phrase `includes next record`, the next log record that is printed is included within the CLR log record as the compensating operation. Otherwise, you must assume that the compensating operation is the logical undo of the log record to which the **link** field of the CLR points.

Checkpoints with Active Transactions

If any transactions are active at the time of a checkpoint, checkpoint records include subentries that describe each of the active transactions using the following columns:

- Log begin (decimal format)
- Transaction ID (decimal format)
- Unique log number (decimal format)
- Log position (hexadecimal format)
- User name

Distributed Transactions

When distributed transactions (transactions that span multiple database servers) generate log records, they are slightly different than nondistributed transactions. You might need to read and interpret them to determine the state of the transaction on both database servers if a failure occurs as a transaction was committing.

The following log records are involved in distributed transactions:

- BEGPREP
- ENDTRANS
- HEURTX
- PREPARE
- TABLOCKS

For more information about this type of logical-log record, see the material on two-phase commit and logical-log records in the *IBM Informix Administrator's Guide*.

If you are performing distributed transactions with TP/XA, the database server uses an XAPREPARE record instead of a PREPARE record.

Logical-Log Record Structure

Each logical-log record has *header* information. Depending on the record type, additional columns of information also appear in the output, as explained in "Logical-Log Record Types and Additional Columns" on page 4-3.

Logical-Log Record Header

Table 4-1 contains sample output to illustrate the header columns that display for a logical-log record.

Table 4-1. Sample Output from onlog

addr	len	type	xid	id	link
2c018	32	BEGIN	6	3	0
2c038	140	HDELETE	6	0	2c018
2c0c4	64	DELITEM	6	0	2c038
2c104	40	DELITEM	6	0	2c0c4
2c12c	72	HDELETE	6	0	2c104
2c174	44	DELITEM	6	0	2c12c
2c1a0	72	HDELETE	6	0	2c174
2c1e8	44	DELITEM	6	0	2c1a0
2c214	64	HDELETE	6	0	2c1e8
2c254	56	DELITEM	6	0	2c214
2c28c	48	DELITEM	6	0	2c254
2c2bc	24	PERASE	6	0	2c28c
2c2d4	20	BEGCOM	6	0	2c2bc
2c2e8	24	ERASE	6	0	2c2d4
2c300	28	CHFREE	6	0	2c2e8
2c31c	24	COMMIT	6	0	2c300

Table 4-2 defines the contents of each header column.

Table 4-2. Definition of onlog Header Columns

Header Field	Contents	Format
addr	Log-record address (log position)	Hexadecimal
len	Record length in bytes	Decimal
type	Record-type name	ASCII
xid	Transaction number	Decimal
id	Logical-log number	Decimal
link	Link to the previous record in the transaction	Hexadecimal

Logical-Log Record Types and Additional Columns

In addition to the six header columns that display for every record, some record types display additional columns of information. The information that appears varies, depending on record type. Table 4-3 on page 4-4 lists all the record types and their additional columns.

The **Action** column indicates the type of database server action that generated the log entry. The **Additional Columns** and **Format** columns describe what information appears for each record type in addition to the header described in "Logical-Log Record Header" on page 4-3.

Table 4-3. Logical-Log Record Types

Record Type	Action	Additional Columns	Format
ADDCHK	Add chunk.	chunk number	Decimal
		chunk name	ASCII
ADDDBS	Add dbspace.	dbspace name	ASCII
ADDITEM	Add item to index.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		logical page	Decimal
		key number	Decimal
		key length	Decimal
ADDLOG	Add log.	log number	Decimal
		log size (pages)	Decimal
		pageno	Hexadecimal
ALLOCGENPG	Allocate a generic page.	tblspace ID	Decimal
		rowid	Decimal
		slot flags and length	Decimal
		page version if delete	Decimal
		flags, vimage record	Decimal
		rowid for previous	Decimal
		data	ASCII
ALTERDONE	Alter of fragment complete.	tblspace ID	Hexadecimal
		physical page number previous page	Hexadecimal
		logical page number	Decimal
		version of alter	Decimal
ALTSPCOLSNEW	Changed columns in an alter table.	number of columns	Decimal
		special column list	array
ALTSPCOLSOLD	Changed columns in an alter table.	number of columns	Decimal
		special column list	array
BADIDX	Bad index	tblspace ID	Hexadecimal
BEGCOM	Begin commit.	(None)	(None)
BEGIN	Begin work.	date	Decimal
		time	Decimal
		SID	Decimal
		user	ASCII
BEGPREP	Written by the coordinator database server to record the start of the two-phase commit protocol.	flags	Decimal (Value is 0 in a distributed transaction.)
		number of participants	Decimal

Table 4-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
BEGWORK	Begin a transaction.	begin transaction time	Decimal
		user ID	Decimal
		process ID	Decimal
BFRMAP	Simple-large-object free-map change.	tblspace ID	Hexadecimal
		bpageno	Hexadecimal
		status	USED/FREE
		log ID	Decimal
		prev page	Hexadecimal
BLDCL	Build tblspace.	tblspace ID	Hexadecimal
		fextsize	Decimal
		nextsize	Decimal
		row size	Decimal
		ncolumns	Decimal
		table name	ASCII
BMAPFULL	Bitmap modified to prepare for alter.	tblspace ID	Hexadecimal
		bitmap page num	Decimal
BMAP2TO4	2-bit bitmap altered to two 4-bit bitmaps.	tblspace ID	Hexadecimal
		2-bit bitmap page number	Decimal
		flags	Decimal
BSPADD	Add blobspace.	blobspace name	ASCII
BTCPYBCK	Copy back child key to parent.	tblspace ID	Hexadecimal
		parent logical page	Decimal
		child logical page	Decimal
		slot	Decimal
		rowoff	Decimal
		key number	Decimal
BTMERGE	Merge B-tree nodes.	tblspace ID	Hexadecimal
		parent logical page	Decimal
		left logical page	Decimal
		right logical page	Decimal
		left slot	Decimal
		left rowoff	Decimal
		right slot	Decimal
		right rowoff	Decimal
		key number	Decimal

Table 4-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
BTSHUFFL	Shuffle B-tree nodes.	tblspace ID	Hexadecimal
		parent logical page	Decimal
		left logical page	Decimal
		right logical page	Decimal
		left slot	Decimal
		left rowoff	Decimal
		key number	Decimal
		flags	Hexadecimal
BTSPLIT	Split B-tree node.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		parent logical page	Decimal
		left logical page	Decimal
		right logical page	Decimal
		infinity logical page	Decimal
		rootleft logical page	Decimal
		midsplit	Decimal
		key number	Decimal
		key length	Decimal
CDINDEX	Create detached index.	database name	ASCII
		owner	ASCII
		table name	ASCII
		index name	ASCII
CDR	Captures the set of table columns modified by an update statement such as a *bitvector*. This log record allows Enterprise Replication to capture only the changed data to avoid transmitting the unchanged columns to a target site. In the example, the first six columns of the table are unchanged (6 leftmost bits in the **bitvector** are 0), the seventh and eighth columns have been updated (seventh and eighth bits are 1), and so on. The onlog output displays as many bits of bitvector as fit in a single line of the output. To see the entire **bitvector** displayed in hexadecimal, use the **onlog -l** command.	name of CDR record	ASCII
		partition number	Hexadecimal
		bitvector	Binary

	Sample **onlog** output for CDR log record:
	`adr len  type xid id link  name    partno  bitvector`
	`40  36  CDR  14  0  18   UPDCOLS   10009a  000000110100110100`		

CHALLOC	Chunk extent allocation.	pageno	Hexadecimal
		size	Hexadecimal
CHCOMBINE	Chunk extent combine.	pageno	Hexadecimal

Table 4-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
CHFREE	Chunk extent free.	pageno	Hexadecimal
		size	Hexadecimal
CHKADJUP	Update chunk adjunct on disk. The database server writes this record when it moves space from the reserved area to the metadata or user-data area or when the user adds an sbspace chunk.	chunk number	Integer
		ud1_start_page	Integer
		ud1_size	Integer
		md_start_page	Integer
		md_size	Integer
		ud2_start_page	Integer
		ud2_size	Integer
		flags	Hexadecimal
CHPHYLOG	Change physical-log location.	pageno	Hexadecimal
		size in kilobytes	Hexadecimal
		dbspace name	ASCII
CHRESERV	Reserve extent for metadata stealing. This record is written when you add an sbspace chunk.	chunk number	Integer
		page number	Integer
		length	Integer
CHSPLIT	Chunk extent split.	pageno	Hexadecimal
CINDEX	Create index.	tblspace ID	Hexadecimal
		low rowid	Decimal
		high rowid	Decimal
		index descriptor	ASCII
COARSELOCK	Coarse-grain locking	tblspace ID	Hexadecimal
		old coarse-locking flag value	Decimal
		new coarse-locking flag value	Decimal
CKPOINT	Checkpoint.	max users	Decimal
		number of active transactions	Decimal
CLR	Compensation-log record; created during a rollback.	(None)	(None)
CLUSIDX	Create clustered index.	tblspace ID	Hexadecimal
		key number	Decimal
COLREPAI	Adjust BYTE, TEXT, or VARCHAR column.	tblspace ID	Hexadecimal
		number of columns adjusted	Decimal
COMMIT	Commit work.	date	Decimal
		time	Decimal
COMTAB	Compact slot table on a page.	logical page number	Decimal
		number slots moved	Decimal
		compressed slot pairs	ASCII

Table 4-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
COMWORK	End a transaction and commit work.	end transaction time	Decimal
		begin transaction time	Decimal
DELETE	Delete before-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
DELITEM	Delete item from index.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		logical page	Decimal
		key number	Decimal
		key length	Decimal
DERASE	Drop tblspace in down dbspace.	tblspace number	Hexadecimal
		table lock number	Decimal
DINDEX	Drop index.	tblspace ID	Hexadecimal
		key number	Decimal
DPT	List all dirty pages not flushed to disk during a fuzzy checkpoint. This record is written just before the CKPOINT record and linked to it. The DPT records are not written during a full checkpoint because all the dirty pages are flushed to disk	number of dirty pages	Hexadecimal
DRPBSP	Drop blobspace.	blobspace name	ASCII
DRPCHK	Drop chunk.	chunk number	Decimal
		chunk name	ASCII
DRPDBS	Drop dbspace.	dbspace name	ASCII
DRPLOG	Drop log.	log number	Decimal
		log size (pages)	Decimal
		pageno	Hexadecimal
ENDTRANS	Written by both the coordinator and participant database servers to record the end of the transaction. ENDTRANS instructs the database server to remove the transaction entry from its shared-memory transaction table and close the transaction. In the coordinator logical log, each BEGPREP that results in a committed transaction is paired with an ENDTRANS record. If the final decision of the coordinator is to roll back the transaction, no ENDTRANS record is written. In the participant logical log, each ENDTRANS record is paired with a corresponding HEURTX record.	(None)	(None)
ERASE	Drop tblspace.	tblspace ID	Hexadecimal

Table 4-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
FREE_RE	Allocate extent from reserve extent to metadata or user-data area of an sbspace chunk.	chunk number	Integer
		page number	Integer
		length	Integer
		flag	Hexadecimal
HDELETE	Delete home row.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
HEURTX	Written by a participant database server to record a heuristic decision to roll back the transaction. It should be associated with a standard ROLLBACK record indicating that the transaction was rolled back.	flag	Hexadecimal (Value is always 1.)
HINSERT	Home row insert.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
HUPAFT	Home row update, after-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
HUPBEF	Home row update, before-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
HUPDATE	If the home row update before-images and after-images can both fit into a single page, the database server writes a single HUPDATE record.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		forward ptr rowid	Hexadecimal
		old slotlen	Decimal
		new slotlen	Decimal
		number of pieces	Decimal
IDXFLAGS	Index flags.	tblspace ID	Hexadecimal
		key number	Hexadecimal
INSERT	Insert after-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
ISOSPCOMMIT	Log an isolated save-point commit.	end transaction time	Decimal
		begin transaction time	Decimal
LCKLVL	Locking mode (page or row).	tblspace ID	Hexadecimal
		old lockmode	Hexadecimal
		new lockmode	Hexadecimal

Table 4-3. Logical-Log Record Types (continued)

	Record Type	Action	Additional Columns	Format
	LG_ADDBPOOL	Add a buffer pool online.	page size in bytes	Decimal
			number of buffers in the pool	Decimal
			number of lru queues	Decimal
			percent of lru_max_dirty	Decimal
			percent of lru_min_dirty	Decimal
4	LG_PTRUNCATE	Identifies an intention to truncate a table. The partitions are marked to be dropped or reused, according to the specified command option.	tblspace ID	Hexadecimal
4	LG_TRUNCATE	TRUNCATE has freed the extents and the transaction will be committed.	tblspace ID	Hexadecimal
	MVIDXND	Index node moved to allow for 2-bit to 4-bit bitmap conversion.	tblspace ID	Hexadecimal
			old page number	Decimal
			new page number	Decimal
			parent page number	Decimal
			parent slot number	Decimal
			parent slot offset	Decimal
			key number	Decimal
	PBDELETE	Delete tblspace blobpage.	bpageno	Hexadecimal
			status	USED/FREE
			unique ID	Decimal
	PBINSERT	Insert tblspace blobpage.	bpageno	Hexadecimal
			tblspace ID	Hexadecimal
			rowid	Hexadecimal
			slotlen	Decimal
			pbrowid	Hexadecimal
	PDINDEX	Predrop index.	tblspace ID	Hexadecimal
	PGALTER	Page altered in place.	tblspace ID	Hexadecimal
			physical page number	Hexadecimal
	PGMODE	Page mode modified in bitmap.	tblspace ID	Hexadecimal
			logical page number	Decimal
			old mode	Hexadecimal
			new mode	Hexadecimal
	PERASE	Preerase old file. Mark a table that is to be dropped. The database server frees the space on the commit.	tblspace ID	Hexadecimal
	PNGPALIGN8	Use the pages in this tblspace as generic pages.	None	
	PNLOCKID	Change tblspaces lockid.	tblspace ID	Hexadecimal
			old lock ID	Hexadecimal
			new lock ID	Hexadecimal

Table 4-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
PNSIZES	Set tblspace extent sizes.	tblspace ID	Hexadecimal
		fextsize	Decimal
		nextsize	Decimal
PREPARE	Written by a participant database server to record the ability of the participant to commit the transaction, if so instructed.	DBSERVERNAME of coordinator	ASCII
PTADESC	Add alter description information.	tblspace ID	Hexadecimal
		physical page number of previous page	Hexadecimal
		logical page number	Decimal
		number of columns added	Decimal
PTALTER	Alter of fragment begun.	tblspace ID	Hexadecimal
		physical page number previous page	Hexadecimal
		logical page number	Decimal
		alter desc page number	Decimal
		num columns added	Decimal
		version of alter	Decimal
		added rowsize	Decimal
PTALTNEWKEYD	Update key descriptors in a tblspace header after an alter table command.	bytes in key descriptor	Decimal
		data in key descriptor	ASCII
PTALTOLDKEYD	Update key descriptors after an alter table command.	bytes in key descriptor	Decimal
		data in key descriptor	ASCII
PTCOLUMN	Add special columns to fragment.	tblspace ID	Hexadecimal
		number of columns	Decimal
PTEXTEND	Tblspace extend.	tblspace ID	Hexadecimal
		last logical page	Decimal
		first physical page	Hexadecimal
PTRENAME	Rename table.	tblspace ID	Hexadecimal
		old table name	ASCII
		new table name	ASCII
RDELETE	Remainder page delete.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
RENDBS	Rename dbspace.	new dbspace name	ASCII
REVERT	Logs the reversion of a database space to a database space of an earlier version.	type of reversion event	Decimal
		arg1	Decimal
		arg2	Decimal
		arg3	Decimal

Table 4-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
RINSERT	Remainder page insert.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
ROLLBACK	Rollback work.	date	Decimal
		time	Decimal
ROLWORK	End a transaction and roll back work.	end transaction time	Decimal
		begin transaction time	Decimal
RSVEXTEND	Logs the extension to the reserved pages.	number of pages	Decimal
		physical page number of extent	Hexadecimal
RTREE	Logs inserts and deletions for R-tree index pages. (Other operations on R-tree indexes are physically logged.) The record subtypes are: • LEAFINS - insert item in a leaf page • LEAFDEL - delete item from leaf page	record subtype	ASCII
		[index page rowid	Hexadecimal
		tuple length	Decimal
		base table rowid	Decimal
		base table fragid	Decimal
		delete flag]	Decimal
RUPAFT	Remainder page update, after-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
RUPBEF	Remainder page update, before-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		slotlen	Decimal
RUPDATE	If the remainder page update before-images and after-images can both fit into a single page, the database server writes a single RUPDATE record.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
		forward ptr rowid	Hexadecimal
		old slotlen	Decimal
		new slotlen	Decimal
		number of pieces	Decimal

Table 4-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
SBLOB	Indicates a subsystem log record for a smart large object. The various record subtypes are: CHALLOC CHCOMBINE CHFREE CHSPLIT CREATE DELETES EXTEND HDRUPD PDELETE PTRUNC REFCOUNT UDINSERT UDINSERT_LT UDUPAFT UDUPAFT_LT UDUPAFT UDUPAFT_LT UDWRITE UDWRITE_LT	Varies For more information, see “Log Record Types for Smart Large Objects” on page 4-14.	Varies
SYNC	Written to a logical-log file if that log file is empty and administrator instructs the database server to switch to next log file.	(None)	(None)
TABLOCKS	Written by either a coordinator or a participant database server. It is associated with either a BEGPREP or a PREPARE record and contains a list of the locked tblspaces (by tblspace number) held by the transaction. (In a distributed transaction, transactions are shown as the owners of locks.)	number of locks	Decimal
		tblspace number	Hexadecimal
UDINSERT	Append new user data.	chunk	Decimal
		page within chunk	Hexadecimal
		offset within page	Hexadecimal
		data length	Hexadecimal
UDUPAFT	Update user data after-image if a UDWRITE is too expensive.	chunk	Decimal
		page within chunk	Hexadecimal
		offset within page	Hexadecimal
		data length	Hexadecimal
UDUPBEF	Update user-data before-image if a UDWRITE is too expensive.	chunk	Decimal
		page within chunk	Hexadecimal
		offset within page	Hexadecimal
		data length	Hexadecimal

Table 4-3. Logical-Log Record Types (continued)

Record Type	Action	Additional Columns	Format
UDWRITE	Update user data (difference image).	chunk	Decimal
		page within chunk	Hexadecimal
		offset within chunk	Hexadecimal
		length before write	Hexadecimal
		length after write	Hexadecimal
UNDO	Header record to a series of transactions to be rolled back.	count	Decimal
UNDOBLDC	This record is written if a CREATE TABLE statement should be rolled back but cannot be because the relevant chunk is down. When the log file is replayed, the table will be dropped.	tblspace number	Hexadecimal
UNIQID	Logged when a new serial value is assigned to a row.	tblspace ID	Hexadecimal
		unique ID	Decimal
UPDAFT	Update after-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
UPDBEF	Update before-image.	tblspace ID	Hexadecimal
		rowid	Hexadecimal
XAPREPARE	Participant can commit this XA transaction.	(None)	(None)

Log Record Types for Smart Large Objects

All smart-large-object log records are the SBLOB type. Each smart-large-object log record contains six header columns, described in "Logical-Log Record Header" on page 4-3; the record subtype; and additional information. The information that appears varies, depending on record subtype.

Table 4-4 lists all the smart-large-object record types. The **Subtype** column describes the smart-large-object record type. The **Action** column indicates the type of database server action that generated the log entry. The **Additional Columns** and **Format** columns describe what information appears for each record type.

Table 4-4. Record Subtypes for Smart Large Objects

Record Subtype	Action	Additional Columns	Format
CHALLOC	Allocate chunk extent.	extent [chk, page, len]	Decimal
		flags	Hexadecimal
CHCOMBINE	Combine two pages in the user-data extent list.	chunk number	Decimal
		first page	Decimal
		second page	Decimal
CHFREE	Frees chunk extent.	extent [chk, page, len]	Decimal
CHSPLIT	Split a page in the user-data extent list.	chunk number	Decimal
		UDFET page to split	Decimal

Table 4-4. Record Subtypes for Smart Large Objects (continued)

Record Subtype	Action	Additional Columns	Format
CREATE	Create smart large object.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		number of extents in lomaphdr	Decimal
DELETE	Delete a smart large object at commit.	smart-large-object ID [sbs, chk, page, oid]	Decimal
EXTEND	Add extent to an extent list of a smart large object.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		extent [chk, page, len]	Decimal
		lomap overflow page number	Decimal
HDRUPD	Update smart-large-object header page.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		old EOF offset	String
		new EOF offset	String
		old times	Decimal
		new times	Decimal
PDELETE	Queue a smart large object for deletion at commit.	smart-large-object ID [sbs, chk, page, oid]	Decimal
PTRUNC	Queue a smart large object for truncation at commit.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		old offset	String
		new offset	String
REFCOUNT	Increment or decrement the reference count of a smart large object.	smart-large-object ID [sbs, chk, page, oid]	Decimal
		1 if increment; 0 if decrement	Decimal
UDINSERT,	Append new user data.	chunk	Decimal
UDINSERT_LT		page within chunk	Decimal
		offset within page	Decimal
		data length	Decimal
UDUPAFT,	Update user-data after-image if a UDWRITE is too expensive.	chunk	Decimal
UDUPAFT_LT		page within chunk	Decimal
		offset within page	Decimal
		data length	Decimal
UDUPBEF,	Update user-data beforeimage if a UDWRITE is too expensive.	chunk	Decimal
UDUPBEF_LT		page within chunk	Decimal
		offset within page	Decimal
		data length	Decimal

Table 4-4. Record Subtypes for Smart Large Objects (continued)

Record Subtype	Action	Additional Columns	Format
UDWRITE,	Update user data (difference image).	chunk	Decimal
UDWRITE_LT		page within chunk	Decimal
		offset within page	Decimal
		length before write	Decimal
		length after write	Decimal
		number of different image pieces	Decimal

For an example of smart-large-object records in **onlog** output, see smart-large-object log records in the chapter on what is the logical log in the *IBM Informix Administrator's Guide*.

Figure 4-1 shows an example of smart-large-object records in **onlog** output. The first two records show that an extent was freed. The next group of records, flanked by BEGIN and COMMIT, shows the allocation of storage and creation of the smart large objects.

```
addr    len   type   xid      id  link     subtype  specific-info

4e8428   40   SBLOB    8       0  4e7400   CHFREE   (2,53,421)
4e8450   40   SBLOB    8       0  4e8428   CHFREE   (2,579,421)

c8018    40   BEGIN    8       3  0        07/13/98 10:23:04 34       informix
c8040    264  SBLOB    8       0  c8018    CREATE   [2,2,1,900350517]  10
c8148    44   SBLOB    8       0  c8040    CHALLOC  (2,53,8) 0x1
c8174    68   SBLOB    8       0  c8148    EXTEND   [2,2,1,900350517] (2,53,8) -1
c81b8    264  SBLOB    8       0  c8174    CREATE   [2,2,2,900350518]  10
c82c0    44   SBLOB    8       0  c81b8    CHALLOC  (2,61,1) 0x1
c82ec    68   SBLOB    8       0  c82c0    EXTEND   [2,2,2,900350518] (2,61,1) -1
c8330    56   SBLOB    8       0  c82ec    REFCOUNT [2,2,1,900350517] 1
c8368    56   SBLOB    8       0  c8330    REFCOUNT [2,2,2,900350518] 1
c83a0    36   COMMIT   8       0  c8368    07/13/98 10:23:05

c83c4    40   BEGIN    8       3  0        07/13/98 10:23:05 34       informix
c83ec    264  SBLOB    8       0  c83c4    CREATE   [2,2,3,900350519]  10
c84f4    44   SBLOB    8       0  c83ec    CHALLOC  (2,62,1) 0x1
c8520    68   SBLOB    8       0  c84f4    EXTEND   [2,2,3,900350519] (2,62,1) -1
c8564    56   SBLOB    8       0  c8520    REFCOUNT [2,2,3,900350519] 1
c859c    36   COMMIT   8       0  c8564    07/13/98 10:23:05
```

Figure 4-1. Smart-Large-Object Records in onlog Output

Part 2. Administrative Utilities

Chapter 5. Overview of Utilities

In This Chapter

This chapter provides reference material for the Informix database server utilities. These utilities allow you to perform administrative tasks directly from the command line. For a complete listing of utilities, see your *IBM Informix Getting Started Guide*.

You can use the following utilities:

- IBM Informix Server Administrator (ISA)
- ON–Bar
- **oncheck**
- **ondblog**
- **oninit**
- **onlog**
- **onmode**
- ON–Monitor
- **onparams**
- **onspaces**
- **onstat**
- **ontape**

The database server must be online before you execute a utility, with the following exceptions:

- **oninit**
- Some **onlog** options
- Some **oncheck** options

Note: When using utilities, do not use the UNIX command `CTRL-C` to send an interrupt signal to a process because it might produce an error.

Complete List of Utilities

The appendix in your *IBM Informix Getting Started Guide* contains a quick reference to all utilities and their options.

Obtaining Utility Version Information

All Informix command-line utilities allow you to obtain version information using **-V** and **-version** options. You use the **-V** and **-version** options primarily for debugging. When a Technical Support representative asks for the version number, you can use the **-V** and **-version** options to find the information.

The **-V** option displays the software version number and the serial number.

The **-version**, extends the **-V** option to display additional information on the build operation system, build number, and build date.

Syntax of Utility-Specific Options

The following syntax diagram illustrates the **-V** and **-version** options

```
►►—utility——┬utility specific options┬——————————————————————————————►◄
            ├-V——————————————————————┤
            └-version————————————————┘
```

The **-V** and **-version** options cannot be used with any other utility options. For example, the **onstat -version** command might display the following output.

```
onstat -version

        Program:          onstat
    Build Version:    10.00.UC1
    Build Host:       connla
    Build OS:         SunOS 5.6
    Build Number:     009
    Build Date:       Sat Aug 12 03:38:27 CDT 2003
    GLS Version:      glslib-4.00.UC2
```

The **onstat -V** command might display the following information:

```
    IBM Informix Dynamic Server Version 10.00.UC1   Software Serial Number
 RDS#N000000
```

Multibyte Characters (GLS)

The database server utilities support multibyte command-line arguments. For a complete list of the utilities that support multibyte command-line arguments, see the *IBM Informix GLS User's Guide*.

IBM Informix Server Administrator

IBM Informix Server Administrator (ISA) allows a DBA to manage Informix database servers by executing Informix commands from any web browser. You do not need to be familiar with the syntax and format of database server commands. ISA presents the command output in an easy-to-read format.

The database server CD-ROM distributed with your product includes ISA. For information on how to install ISA, see the following file on the CD-ROM.

Operating System	File
UNIX	**/SVR_ADM/README**
Windows	**\SVR_ADM\readme.txt**

With ISA, you can perform these database server administrative tasks:
- Change configuration parameters temporarily or permanently.
- Use **Server Setup** to configure or reconfigure the database server.
- Change the database server mode.

- Modify connectivity information in the **sqlhosts** file.
- Check dbspaces, blobspaces, and sbspaces.
- Manage logical logs and physical logs.
- Examine and modify memory usage.
- Read the message log.
- Back up and restore dbspaces, blobspaces, and sbspaces.
- Run various **onstat** commands to monitor performance.
- Enter SQL statements and examine database schemas.
- Add and remove chunks, dbspaces, blobspaces, sbspaces.
- Examine and manage user sessions.
- Examine and manage virtual processors (VPs).
- Use the High-Performance Loader (HPL), **dbimport**, and **dbexport**.
- Manage Enterprise Replication.
- Manage a MaxConnect server.
- Set up primary and secondary database servers for High-Availability Data Replication.
- Use the following utilities: **dbaccess**, **dbschema**, **onbar**, **oncheck**, **ondblog**, **oninit**, **onlog**, **onmode**, **onparams**, **onspaces**, **onstat**, **onpladm**.
- Enter any Informix utility, UNIX shell command, or Windows command.

Chapter 6. The oncheck Utility

In This Chapter

Depending on the options that you choose, **oncheck** can perform the following functions:

- Check specified disk structures for inconsistencies.
- Repair indexes that are found to contain inconsistencies.
- Display information about the disk structures.
- Check and display information about user-defined data types across distributed databases.

oncheck Check-and-Repair Options

The **oncheck** utility can repair the following types of disk structures:

- Partition page statistics
- Bitmap pages
- Partition blobpages
- Blobspace blobpages
- Indexes
- Sbspace pages
- Metadata partitions for sbspaces

If **oncheck** detects inconsistencies in other structures, messages alert you to these inconsistencies, but **oncheck** cannot resolve the problem. For more information, see the chapter on consistency checking in the *IBM Informix Administrator's Guide* and Chapter 3, "Disk Structures and Storage," on page 3-1

What Does Each Option Do?

As Table 6-1 on page 6-2 shows, the **oncheck** options fall into three categories: check, repair, and display. The display or print options (those prefixed with the letter **p**) are identical in function to the **-c** options, except that the **-p** options display additional information about the data that is being checked as the **oncheck** utility executes. You cannot combine **oncheck** option flags except as the following paragraphs describe.

In general, the **-c** options check for consistency and display a message on the screen only if they find an error or inconsistency.

Any user can execute the check options. On UNIX platforms, you must be user **informix** or **root** to display database data or initiate repair options. On Windows, you must be a member of the **Informix-Admin** group to display database data or initiate repair options.

Table 6-1 associates **oncheck** options with their function.

Table 6-1. oncheck Options and Their Function

Object	Check	Repair	Display
Blobspace simple large objects			**-pB**
System catalog tables	**-cc**		**-pc**
Data rows, no simple large objects or smart large objects	**-cd**		**-pd**
Data rows, simple large objects but no smart large objects	**-cD**		**-pD**
Table with a user-defined access method	**-cd, -cD**		
Chunks and extents	**-ce**		**-pe**
Index (key values)	**-ci, -cix**	**-ci -y** **-pk -y, -pkx -y**	**-pk**
Index (keys plus rowids)	**-cI, -cIx**	**-cI -y** **-pK -y, -pKx -y**	**-pK**
Index with a user-defined access method	**-ci, -cI**		
Index (leaf key values)		**-pl -y, -plx -y**	**-pl**
Index (leaf keys plus rowids)		**-pL -y, -pLx -y**	**-pL**
Pages (by table or fragment)			**-pp**
Pages (by chunk)			**-pP**
Root reserved pages	**-cr, -cR**		**-pr, -pR**
Metadata for smart large objects	**-cs, -cS**		**-ps, -pS**
Space usage (by table or fragment)			**-pt**
Space usage (by table, with indexes)			**-pT**

Using the -y Option to Perform Repairs

Use the **-y** option to instruct **oncheck** to perform repairs automatically, as the following examples show:

```
oncheck -cd -y
oncheck -cD -y
oncheck -ci -y
oncheck -cI -y
```

If you do not use the **-y** option, **oncheck** prompts you when it encounters an inconsistency and allows you to request a repair. If you specify option **-n**, **oncheck** does not prompt you because this option instructs **oncheck** to not perform repairs.

Repairing Fragmented Tables

The **oncheck** utility cannot repair a table in a dbspace, sbspace, or external space.

Repairing Indexes in Sbspaces and External Spaces

The **oncheck** utility can repair an index in an sbspace or external space if the index is created using an access method that supports the **oncheck -y** option. Although the **oncheck** utility does not repair fragmented indexes, user-defined access methods can repair them. For more information about the **oncheck** options that access methods support, see the *IBM Informix DataBlade API Programmer's Guide* or the *IBM Informix Virtual-Index Interface Programmer's Guide*.

Locking and oncheck

The **oncheck** utility places a shared lock on a table during the following operations, so no other users can perform updates, inserts, or deletes until the check has completed:

- When it checks data
- When it checks indexes (with **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL**) and the table uses page locking
- When you specify the **-x** option with **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL** and the table uses row locking

If the table does not use page locking, the database server does not place a shared lock on the table when you check an index with the **oncheck -ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL** options. When no shared lock is on the table during an index check, other users can update rows during the check.

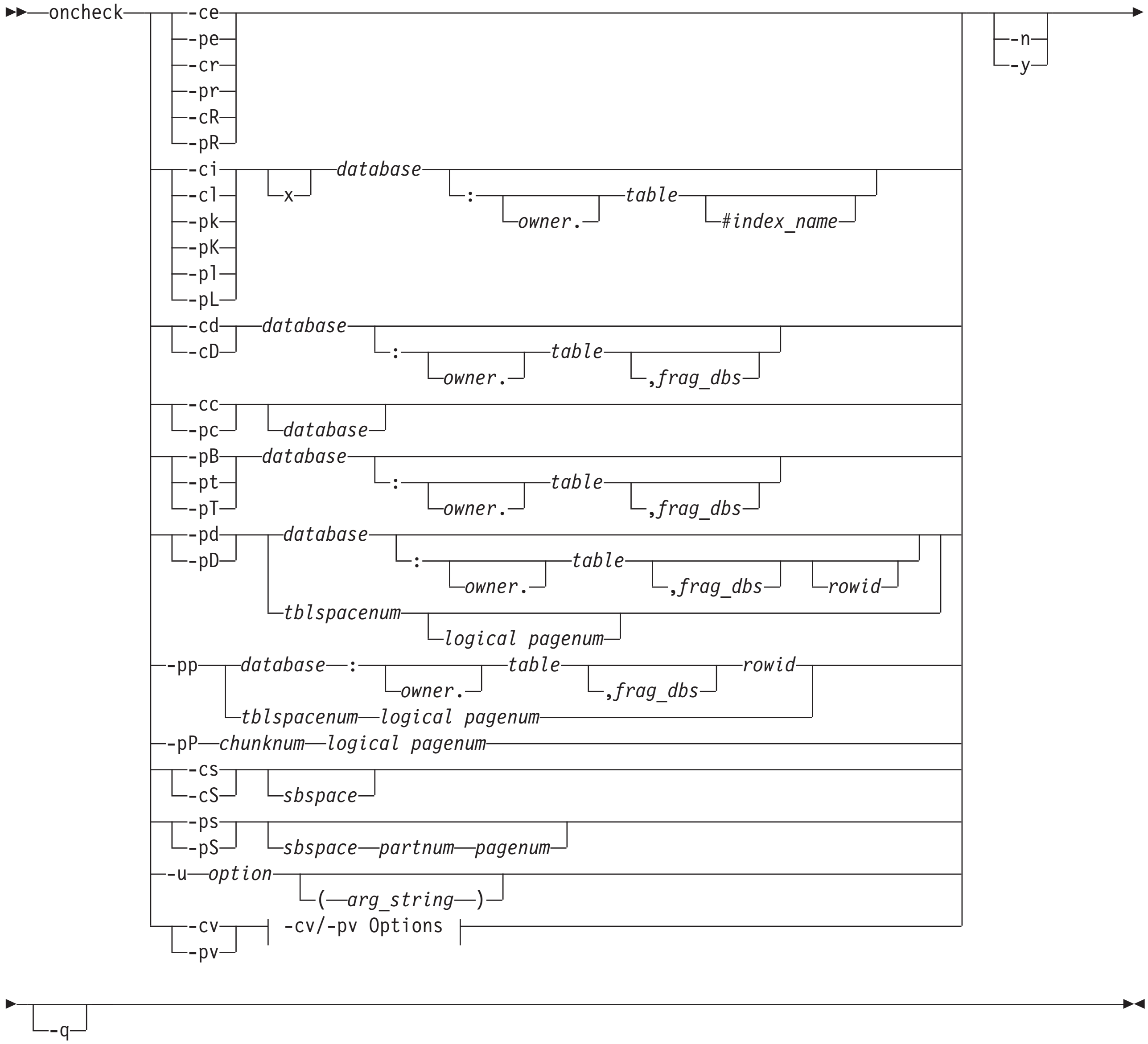
By not placing a shared lock on tables using row locks during index checks, the **oncheck** utility cannot be as accurate in the index check. For absolute assurance of a complete index check, you can execute **oncheck** with the **-x** option. With the **-x** option, **oncheck** places a shared lock on the table, and no other users can perform updates, inserts, or deletes until the check has completed.

For more information about the **-x** option, refer to "Turn On Locking with -x" on page 6-16. For information on shared locks and intent shared locks, see the *IBM Informix Performance Guide*.

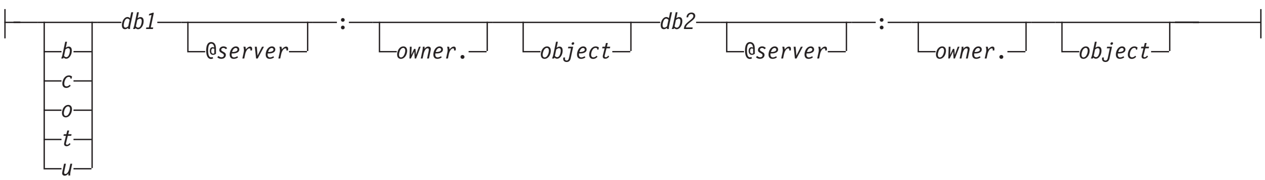
The **oncheck** utility places a shared lock on system catalog tables when they are checked. It places an exclusive lock on a table when it executes repair options.

Syntax

oncheck Options



-cv/-pv Options:



Element	Purpose	Key Considerations
-cc	Checks system catalog tables for the specified database	**References:** See "Check System Catalog Tables with -cc" on page 6-8.

Element	Purpose	Key Considerations
-cd	Reads all pages except simple large objects from the tblspace for the specified database, table, or fragment and checks each page for consistency Also checks tables that use a user-defined access method	**Restrictions:** Does not check simple or smart large objects. **References:** See "Check Pages with -cd and -cD" on page 6-8.
-cD	Same as -**cd** but also reads the header of each blobpage and checks it for consistency	**Restrictions:** Checks simple large objects but not smart large objects. **References:** See "Check Pages with -cd and -cD" on page 6-8.
-ce	Checks each chunk-free list and corresponding free space and each tblspace extent. Also checks smart-large-object extents and sbspace metadata	**Additional Information**: The **oncheck** process verifies that the extents on disk correspond to the current control information that describes them. **References**: See "Check the Chunk Free List with -ce and -pe" on page 6-10. For background information, see "Next-Extent Allocation" on page 3-10.
-ci	Checks the ordering of key values and the consistency of horizontal and vertical node links for all indexes associated with the specified table Also checks indexes that use a user-defined access method	**References:** See "Check Index Node Links with -ci and -cI" on page 6-10.
-cI	Same as -**ci** but also checks that the key value tied to a rowid in an index is the same as the key value in the row	**References:** See "Check Index Node Links with -ci and -cI" on page 6-10.
-cr	Checks each of the root dbspace reserved pages for several conditions	**References:** See "Check Reserved Pages with -cr and -cR" on page 6-11.
-cR	Checks the root dbspace reserved pages, physical-log pages, and logical-log pages	None.
-cs	Checks smart large object and sbspace metadata for an sbspace	**References:** See "Check and Display Sbspaces with -cs, -cS, -ps, -pS" on page 6-12.
-cS	Checks smart large object and sbspace metadata for an sbspace as well as extents	**References:** See "Check and Display Sbspaces with -cs, -cS, -ps, -pS" on page 6-12.
sbspace	Indicates optional sbspace name If not supplied, all sbspaces are checked.	None.
-n	Indicates that no index repair should be performed, even if errors are detected	**Additional Information**: Use with the index repair options (**-ci**, **-cI**, **-pk**, **-pK**, **-pl**, and **-pL**).

Element	Purpose	Key Considerations
-pB	Displays statistics that describe the average fullness of blobspace blobpages in a specified table	**Additional Information**: These statistics provide a measure of storage efficiency for individual simple large objects in a database or table. If a table or fragment is not specified, statistics are displayed for the entire database. **References**: See "Display Blobspace Statistics with -pB" on page 6-12. For information about optimizing blobspace blobpage size, see the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-pc	Same as **-cc** but also displays the system catalog information as it checks the system catalog tables, including extent use for each table	None.
-pd	Displays rows in hexadecimal format	**References:** See "Display Rows in Hexadecimal Format with -pd and -pD" on page 6-12.
-pD	Displays rows in hexadecimal format and simple-large-object values stored in the tblspace or header information for smart large objects stored in an sbspace sbpage and simple large objects stored in a blobspace blobpage	**References:** See "Display Rows in Hexadecimal Format with -pd and -pD" on page 6-12.
-pe	Same as **-ce** but also displays the chunk and tblspace extent information as it checks the chunk free list, the corresponding free space, and each tblspace extent	None.
-pk	Same as **-ci** but also displays the key values for all indexes on the specified table as it checks them	**References:** See "Display Index Information with -pk, -pK, -pl, -pL" on page 6-13.
-pK	Same as **-cI** but also displays the key values and rowids as it checks them	**References:** See "Display Index Information with -pk, -pK, -pl, -pL" on page 6-13.
-pl	Same as **-ci** but also displays the key values. Only leaf-node index pages are checked	**References:** See "Display Index Information with -pk, -pK, -pl, -pL" on page 6-13.
-pL	Same as **-cI** but also displays the key values and rowids for leaf-node index pages only	**References:** See "Display Index Information with -pk, -pK, -pl, -pL" on page 6-13.
-pp	Displays contents of a logical page	**References:** See "Display the Contents of a Logical Page with -pp and -pP" on page 6-14.
-pP	Same as **-pp** but requires a chunk number and logical page number or internal rowid as input	**References:** See "Display the Contents of a Logical Page with -pp and -pP" on page 6-14.
-pr	Same as **-cr** but also displays the reserved-page information as it checks the reserved pages	**References:** See "Display Reserved-Page Information with -pr and -pR" on page 6-15.
-pR	Same as **-cR** but also displays the information for the reserved pages, physical-log pages, and logical-log pages	None.
-ps	Checks and displays smart-large-object and sbspace metadata for an sbspace	**References:** See "Check and Display Sbspaces with -cs, -cS, -ps, -pS" on page 6-12.

Element	Purpose	Key Considerations
-pS	Checks and displays smart-large-object and sbspace metadata. Lists extents and header information for individual smart large objects	**References:** See "Check and Display Sbspaces with -cs, -cS, -ps, -pS" on page 6-12.
-pt	Displays tblspace information for a table or fragment	**References:** See "Display Tblspaces for a Table or Fragment with -pt and -pT" on page 6-15.
-pT	Same as **-pt** but also displays index-specific information and page-allocation information by page type (for dbspaces)	**References:** See "Display Tblspaces for a Table or Fragment with -pt and -pT" on page 6-15.
-q	Suppresses all checking and validation message	None.
-x	Places a shared lock on the table when you check and print an index	**Additional information:** Use with the **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL** options. **References:** See "Turn On Locking with -x" on page 6-16.
-y	Repairs indexes when errors are detected	None.
chunknum	Specifies a decimal value that you use to indicate a particular chunk	**Restrictions**: Value must be an unsigned integer greater than 0. Chunk must exist. **Additional Information**: Execute the -pe option to learn which chunk numbers are associated with specific dbspaces, blobspaces or sbspaces.
database	Specifies the name of a database that you want to check for consistency	**References**: Syntax must conform to the Identifier segment; see *IBM Informix Guide to SQL: Syntax.*
db1	Specifies the local database that contains a data type that you want to check	**Additional Information**: Optionally specify the local database server name using the format **db1@server1**.
db2	Specifies the remote database that contains a data type that you want to check	**Additional Information**: Optionally specify the remote database server name using the format **db2@server2**.
frag_dbs	Specifies the name of a dbspace that contains a fragment you want to check for consistency	**Restrictions**: Dbspace must exist and contain the fragment that you want to check for consistency. **References**: Syntax must conform to the Identifier segment; see *IBM Informix Guide to SQL: Syntax*.
index_name	Specifies the name of the index that you want to check for consistency	**Restrictions**: Index must exist on table and in database specified. **References**: Syntax must conform to the Identifier segment; see *IBM Informix Guide to SQL: Syntax.*
logical pagenum	Specifies an integer value that you use to indicate a particular page in a tblspace	**Restrictions**: Value must be an unsigned integer between 0 and 16,777,215, inclusive. **Additional Information**: Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
object	Specifies the name of the DataBlade, cast, operator class, user-defined data type, or UDR that you want to check	**Additional Information**: If you do not specify an object name, the database server compares all objects of the same type with the same name and owner.

Element	Purpose	Key Considerations
owner	Specifies the owner of a table	**Restrictions**: You must specify the current owner of the table. **References**: Syntax must conform to the Owner Name segment; see *IBM Informix Guide to SQL: Syntax.*
pagenum	Indicates the page number of the sbspace metadata portion to check and display	None.
partnum	Identifies the sbspace metadata partition to check and display	None.
rowid	Identifies the rowid of the row whose contents you want to display. The rowid is displayed as part of **oncheck -pD** output	**Restrictions**: Value must be an unsigned integer between 0 and 4,277,659,295, inclusive. **Additional Information**: Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
sbspace	Specifies the name of the sbspace that you want to check for consistency	None.
server	Specifies the database server name	**Additional Information**: If you omit the database server name, **oncheck** uses the name that INFORMIXSERVER specifies.
table	Specifies the name of the table that you want to check for consistency	**Additional Information**: Table exists when you execute the utility. **References**: Syntax must conform to the Table Name segment; see *IBM Informix Guide to SQL: Syntax.*
tblspacenum	Identifies the tblspace whose contents you want to display	**Restrictions**: Value must be an unsigned integer between 0 and 208,666,624, inclusive. **Additional Information**: Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.

Check System Catalog Tables with -cc

The **-cc** option checks all system catalog tables for the specified database. If you do not specify a database, it checks all system catalog tables for all databases. Before you execute **oncheck**, execute the SQL statement UPDATE STATISTICS to ensure that an accurate check occurs.

To check a table, **oncheck** compares each system catalog table to its corresponding entry in the tblspace. (See "Structure of the Tblspace Tblspace" on page 3-5.) The **-pc** option performs the same checks and also displays the system catalog information, including the physical address, type of locking used, row size, number of keys, extent use, the number of pages allocated and used, tblspace partnum, and index use for each table.

```
oncheck -cc
oncheck -cc superstores_demo
```

Check Pages with -cd and -cD

The **-cd** option reads all pages, excluding blobpages and sbpages, from the tblspace for the specified database, table, or fragment and checks each page for consistency. It checks entries in the bitmap page against the pages to verify mapping.

If the database contains fragmented tables, but you do not specify a fragment, the option checks all fragments in the table. If you do not specify a table, it checks all tables in the database. (The **-pd** option displays a hexadecimal dump of specified pages but does not check for consistency.)

For both the **-cd** and **-cD** options, the **oncheck** utility locks each table as it checks the indexes for the table. To repair the pages, specify **oncheck -cd -y** or **-cD -y**.

The **-cD** option performs checks similar to those performed when you use the **-cd option,** but it includes a consistency check of blobpages. The **-cD** option checks only the header of each blobpage for consistency. Because **oncheck** does not read the entire page, it does not compare beginning time stamps (stored in the header) with ending time stamps (stored at the end of a blobpage). The **-cD -y** option also cleans up orphaned simple large objects in blobspaces (which could occur after a rollback across several log files).

If tables are fragmented on multiple partitions in the same dbspace, the **oncheck -cd** and **oncheck -cD** commands show the partition names. The following example shows typical output for a table that has fragments in multiple partitions in the same dbspace:

```
TBLspace data check for multipart:informix.t1
              Table fragment partition part_1 in DBspace dbs1
              Table fragment partition part_2 in DBspace dbs1
              Table fragment partition part_3 in DBspace dbs1
              Table fragment partition part_4 in DBspace dbs1
              Table fragment partition part_5 in DBspace dbs1
```

To monitor blobspace blobpages, refer to **oncheck -pB**. (See “Display Blobspace Statistics with -pB” on page 6-12).

The following example checks the data rows, including simple large objects and smart large objects, in the **catalog** table:

```
oncheck -cD superstores_demo:catalog
```

If **oncheck** finds an inconsistency, it displays a message similar to the following one:

```
BAD PAGE 2:28: pg_addr 2:28 != bp-> bf_pagenum 2:69
```

The physical address 2:28 represents page 28 of chunk number 2. If **oncheck** finds no inconsistencies, it displays a header similar to the following one for each table that it checks:

```
TBLSPACE data check for stores_demo:informix.customer
```

If you specify a single fragment, **oncheck** displays a single header for that fragment. The **oncheck** utility displays a header similar to the following one for fragmented tables, one per fragment:

```
TBLspace data check for stores_demo:informix.tab1
 Table fragment in DBspace db1
```

If an index that uses an access method provided by a DataBlade module cannot find the access method, you receive the following message:

```
-9845 Access method access_method_name does not exist in database.
Ensure that the DataBlade installation was successful.
```

Check the Chunk Free List with -ce and -pe

The **-ce** option checks each chunk-free list and corresponding free space and each tblspace extent. (See "Next-Extent Allocation" on page 3-10 and "Structure of the Chunk Free-List Page" on page 3-4, respectively.) The **oncheck** process verifies that the extents on disk correspond to the current control information that describes them.

The **-pe** option performs the same checks and also displays the chunk and tblspace extent information during the check.

```
oncheck -ce
oncheck -pe
```

The **-ce** and **-pe** options also check blobspaces, smart-large-object extents, and user-data and metadata information in sbspace chunks. For information about using **oncheck -ce** and **-pe**, see managing disk space in the *IBM Informix Administrator's Guide*.

Check Index Node Links with -ci and -cI

The **-ci** option checks the ordering of key values and the consistency of horizontal and vertical node links for all indexes associated with the specified table. (See "Structure of B-Tree Index Pages" on page 3-16.)

If you do not specify an index, the option checks all indexes. If you do not specify a table, the option checks all tables in the database.

If the option detects inconsistencies, it prompts you for confirmation to repair the problem index. If you specify the **-y** (yes) option, indexes are automatically repaired. If you specify the **-n** (no) option, the problem is reported but not repaired; no prompting occurs.

If **oncheck** does not find inconsistencies, the following message appears:

```
validating indexes......
```

The message displays the names of the indexes that **oncheck** is checking.

Index rebuilding can be time-consuming if you use **oncheck**. Processing is usually faster if you use the SQL statements DROP INDEX and CREATE INDEX to drop the index and re-create it.

The **-cI** option performs the same checks as **-ci**, but it also checks that the key value tied to a rowid in an index is the same as the key value in the row. The same **-ci** repair options are available with **-cI**.

The following example checks all indexes on the **customer** table:

```
oncheck -cI -n stores_demo:customer
```

The following example checks the index **zip_ix** on the **customer** table:

```
oncheck -cI -n stores_demo:customer#zip_ix
```

If indexes are fragmented on multiple partitions in the same dbspace, the **oncheck -ci** and **oncheck -cI** commands show the partition names. The following example show typical output for an index that has fragments in multiple partitions in the same dbspace:

```
Validating indexes for multipart:informix.t1...
                Index idx_t1
                  Index  fragment partition part_1 in DBspace dbs1
                  Index  fragment partition part_2 in DBspace dbs1
                  Index  fragment partition part_3 in DBspace dbs1
                  Index  fragment partition part_4 in DBspace dbs1
                  Index  fragment partition part_5 in DBspace dbs1
```

By default, the database server does not place a shared lock on the table when you check an index with the **oncheck -ci** or **-cI** options unless the table uses page locking. For absolute assurance of a complete index check, you can execute **oncheck** with the **-x** option. With the **-x** option, **oncheck** places a shared lock on the table, and no other users can perform updates, inserts, or deletes until the check has completed. For more information on option **-x**, see "Turn On Locking with -x" on page 6-16.

When you execute **oncheck** on an external index, the user-defined access method is responsible for checking and repairing an index. If an index that employs a user-defined access method cannot find the access method, the database server reports an error. The **oncheck** utility does not repair inconsistencies in external indexes.

Important: If you are using the Verity Text Search DataBlade Module, the **-cI** option performs an index merge instead of the usual operations. IBM recommends that you do not use **oncheck -cI** for a table that contains more than one type of index.

Check Reserved Pages with -cr and -cR

The **-cr** option checks each of the root dbspace reserved pages (see "Reserved Pages" on page 3-3) as follows:

- It validates the contents of the ONCONFIG file with the PAGE_CONFIG reserved page.
- It ensures that all chunks can be opened, that chunks do not overlap, and that chunk sizes are correct.

The following example checks each of the root dbspace reserved pages:

```
oncheck -cr
```

The **-cR** option performs the same operations as the **-cr** option, but it also checks all logical-log and physical-log pages for consistency. The **-cr** option is considerably faster because it does not check the log-file pages.

If you have changed the value of a configuration parameter (either through ISA, **onparams**, **onmonitor**, **onspaces**, or by editing the configuration file), but you have not yet reinitialized shared memory, **oncheck -cr** and **oncheck -cR** detect the inconsistency and return an error message.

If **oncheck -cr** does not display any error messages after you execute it, you can assume that all three items in the preceding list were checked successfully.

Check and Display Sbspaces with -cs, -cS, -ps, -pS

The **-cs** option checks sbspaces. The **-ps** option checks sbspaces and extents. If you do not specify the sbspace name, these options check all sbspaces. The following example checks the sbspace **test_sbspace**:

```
oncheck -cs test_sbspace
```

The **-cS** and **-pS** options validate and display metadata for an sbspace. The **-pS** option also lists extents and header information for smart large objects. The following example checks and displays metadata for **test_sbspace**:

```
oncheck -ps test_sbspace
```

If you specify **rootdbs** as the sbspace name with the **-cs** or **-ps** options, **oncheck** checks the root dbspace.

For information about using **oncheck -cs, -cS, -ps,** and **-pS**, see monitoring sbspaces in the *IBM Informix Administrator's Guide*.

Display Blobspace Statistics with -pB

The **-pB** option displays statistics that describe the average fullness of blobspace blobpages in a specified table. These statistics provide a measure of storage efficiency for individual simple large objects in a database or table. If you do not specify a table or fragment, the option displays statistics for the entire database. (See optimizing blobspace blobpage size in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.)

```
oncheck -pB photo_base:photos
```

Display Rows in Hexadecimal Format with -pd and -pD

The **-pd** option takes a database, a table, a fragment, and a specific rowid or tblspace number and logical page number as input. In every case, **-pd** prints page-header information and displays the specified rows for the database object (database, table, fragment, internal rowid, or page number) that you specify in hexadecimal and ASCII format. No checks for consistency are performed.

If you specify an internal rowid (expressed as a hexadecimal value), the rowid maps to a particular page, and all rows from that page are printed.

If you specify a logical page number (expressed as a decimal), all the rows of the tblspace number with the logical page number are printed.

If you specify a fragment, all the rows in the fragment are printed, with their rowids, forward pointers, and page type.

If you specify a table, all the rows in the table are printed, with their rowids, forward pointers, and page type.

If you specify a database, all the rows in all the tables in the database are printed. TEXT and BYTE column descriptors stored in the data row are printed, but TEXT and BYTE data itself is not.

The **-pD** option prints the same information as **-pd**. In addition, **-pD** prints TEXT and BYTE values stored in the tblspace or header information for simple large objects stored in a blobspace blobpage. The following example show different options for the **oncheck -pd** and **oncheck -pD** commands:

```
oncheck -pd stores_demo:customer,frgmnt1
oncheck -pd stores_demo:customer
oncheck -pD stores_demo:customer 0x101
```

The following example shows a partial output of an **oncheck -pD** command:

```
oncheck -pD multipart:t1 :

TBLspace data check for multipart:informix.t1
                  Table fragment partition part_1 in DBspace dbs1
page_type  rowid    length fwd_ptr
HOME       101      24     0
    0:  0  0  0  a 47 48 49 20 20 20 20 20 20 20 20 20   ....GHI
   16: 20 20 20 20 20 20 20 20
                                       ........
```

Display Index Information with -pk, -pK, -pl, -pL

Repair options are available for each option.

The **-pk** option performs the same checks as the **-ci** option. (See "Check Index Node Links with -ci and -cI" on page 6-10.) In addition, **-pk** displays the key values for all indexes on the specified table as it checks them.

The **-pK** option performs the same checks as the **-cI** option. The **-pK** option displays the key values and rowids as it checks them.

The **-pl** option performs the same checks as the **-ci** option and displays the key values, but it checks only leaf-node index pages. It ignores the root and branch-node pages. See "Structure of B-Tree Index Pages" on page 3-16.

The **-pL** option performs the same checks as the **-cI** option and displays the key values and rowids, but it checks only leaf-node index pages. It ignores the root and branch-node pages.

```
oncheck -pL -n stores_demo.customer
```

The following example displays information about all indexes on the **customer** table:

```
oncheck -pl -n stores_demo:customer
```

The following example displays information about the index **zip_ix**, which was created on the **customer** table:

```
oncheck -pl -n stores_demo:customer#zip_ix
```

By default, the database server does not place a shared lock on the table when you check an index with the **oncheck -pk, -pK, -pl,** or **-pL** options unless the table uses page locking. For absolute assurance of a complete index check, you can execute **oncheck** with the **-x** option. With the **-x** option, **oncheck** places a shared lock on the table, and no other users can perform updates, inserts or deletes until the check has completed. For more information on option **-x**, see "Turn On Locking with -x" on page 6-16.

Display the Contents of a Logical Page with -pp and -pP

The **-pp** option has the following syntax variations:

Invocation	Explanation
oncheck -pp tblspc lpn <pages>	Displays the contents of a logical page using a tablespace number and logical page number. You can also specify an optional parameter specifying the number of pages to be printed.
oncheck -pp *tblspc lpn -h*	Displays only the header of a logical page using a tablespace number and logical page number.
oncheck -pp *database:table rowid*	Displays the contents of a logical page using a database name, table name, and an Informix internal rowid. You can obtain this internal rowid with the **oncheck -pD** command. This internal rowid is not the serial rowid that is assigned in tables created with the CREATE TABLE tabname WITH ROWIDS statement. For more information, see "Definition of Rowid" on page 3-13

The page contents appear in ASCII format. The display also includes the number of slot-table entries on the page. The following example shows different invocations of the **oncheck -pp** command:

```
oncheck -pp stores_demo:orders 0x211  # database:owner.table,   # fragment rowid
oncheck -pp stores_demo:informix.customer,frag_dbspce1 0x211
oncheck -pp 0x100000a 25 # specify the tblspace number and # logical page number
```

The **-pP** option provides the following syntax variations:

Invocation	Explanation
oncheck -pP chunk# offset pages	Displays the contents of a logical page using a chunk number and an offset. You can also specify an optional parameter specifying the number of pages to be printed.
oncheck -pP *chunk# offset -h*	Displays only the header of a logical page using a chunk number and an offset.

Note: The output for chunk page displays both the `start` and the `length` fields in decimal format.

The following example shows typical output using the **onstat -pP** command:

```
oncheck -pP 1 5 2
addr        stamp        nslots        flag      type        frptr      frcnt      next      prev      stamp
100005        250181        2        1000      ROOTRSV        320      1716      0      0      250181
slot      ptr      len      flg
...
addr        stamp        nslots        flag      type        frptr      frcnt      next      prev      stamp
100005   6      250182        2        1000      ROOTRSV        128      1908      0      0      250182
slot      ptr      len      flg
1      24      56      0
2      80      48       0
```

Display Reserved-Page Information with -pr and -pR

The **-pr** and **-pR** options perform the same checks as **oncheck -cr** and **oncheck -cR**, respectively, and also display the reserved-page information. The **-pR** option displays detailed information about logical-log and physical-log pages, marking the start and end of the active physical-log pages.

(For a description of the **-cr** option, see "Check Reserved Pages with -cr and -cR" on page 6-11.) For a listing and explanation of **oncheck -pr** output, see "Reserved Pages" on page 3-3.

```
oncheck -pr
```

If you have changed the value of a configuration parameter (either through ISA or by editing the configuration file), but you have not yet reinitialized shared memory, **oncheck -pr** and **oncheck -pR** detect the inconsistency and return an error message.

Display Tblspaces for a Table or Fragment with -pt and -pT

The **-pt** option prints a tblspace report for a given table or fragment whose name and database you specify when you execute **oncheck** at the command line. The report contains general allocation information including the maximum row size, the number of keys, the number of extents, their sizes, the pages allocated and used per extent, the current serial value, and the date that the table was created. The **-pt** output prints the pagesize of the tablespace, the number of pages (allocated, used and data) in terms of logical pages. The **Extents** fields list the physical address for the tblspace **tblspace** entry for the table and the address of the first page of the first extent. The extent list shows the number of logical as well as physical pages in every extent. If you do not specify a table, the option displays this information for all tables in the database.

The **-pT** option prints the same information as the **-pt** option. In addition, the **-pT** option displays index-specific information and page-allocation information by page type (for dbspaces).

Output for both **-pt** and **-pT** contains listings for **Number of pages used.** The value shown in the output for this field is never decremented because the disk space allocated to a tblspace as part of an extent remains dedicated to that extent even after you free space by deleting rows. For an accurate count of the number of pages currently used, refer to the detailed information on tblspace use (organized by page type) that the **-pT** option provides. The following example shows a typical **oncheck -pT command:**

```
oncheck -pT stores_demo:customer
```

The following example shows an example of output of the **oncheck -pt command:**

```
TBLspace Report for testdb:tab1

Physical Address            2:10
Creation date               10/07/2004 17:01:16
TBLspace Flags              801        Page Locking
                                           TBLspace use 4 bit bit-maps
Maximum row size            14
Number of special columns   0
Number of keys              0
Number of extents           1
Current serial value        1
Pagesize (k)                4
```

```
First extent size              4
Next extent size               4
Number of pages allocated      340
Number of pages used           337
Number of data pages           336
Number of rows                 75806
Partition partnum              2097154
Partition lockid               2097154

Extents
       Logical Page     Physical Page        Size Physical Pages
                 0             2:106         340       680
```

For more examples of using **oncheck -pt** and **-pT**, see managing disk space in the *IBM Informix Administrator's Guide* and the *IBM Informix Performance Guide*.

Turn On Locking with -x

If you append the **-x** option to the index-checking options, **oncheck** places a shared lock on affected tables while it checks the indexes, meaning no other users can perform inserts, updates, and deletions while **oncheck** checks or prints the index. Without the -x option for tables with row locking, **oncheck** only places an IS (intent shared) lock on the table, which prevents actions such as dropping the table or the indexes during the check.

You can append the **-x** option to the **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, and **-pL** options. For example, the following sample command instructs **oncheck** to lock indexes for the **customer** table while it validates the order of key values, validates horizontal links, and ensures that no node appears twice in the index:

```
oncheck -cix stores_demo:customer
```

When you specify option **-x**, **oncheck** locks indexes for tables that use row locking. If **oncheck** detects page-lock mode, it displays a warning message and places a shared lock on the table regardless.

Send Special Arguments to the Access Method with -u

You can use the **-u** option to send special arguments to the access method. The possible arguments depend on the access method. For example, the R-tree access method supports the **display** option, as the following example shows:

```
oncheck -pl -u "display"
```

Use commas to separate multiple arguments in the argument string.

For information on valid arguments for your access method, refer to the user manual for your access method.

Return Codes on Exit

The **oncheck** utility returns the following codes on exit.

```
GLS failures:-1
Invalid srial/key:2
Onconfig access error:2
Invalid onconfig settings:2
Invalid arguments to oncheck:2
Error connecting database server:1
error detected by oncheck:2
no errors detected by oncheck:0
```

Windows only:

```
Not properly installed:1
Authentication error:2
```

Chapter 7. The ondblog Utility

ondblog: Change Logging Mode

The **ondblog** utility lets you change the logging mode for one or more databases.

The **ondblog** utility logs its output in the BAR_ACT_LOG file

If you turn on transaction logging for a database, you must create a level-0 backup of all the storage spaces that contain data in the database before the change takes effect.

For more information and examples, see the following topics in the chapter on managing database-logging status in the *IBM Informix Administrator's Guide*:
- Modifying the database-logging status
- Modifying table-logging status

Syntax

```
>>--ondblog--+-buf----+--+-------------------+----------------------><
             +-unbuf--+  |   .-----------.   |
             +-nolog--+  |   V           |   |
             +-ansi---+  +---+-db_list---+---+
             '-cancel-'  '-----f--dbfile-----'
```

Element	Purpose	Key Considerations
buf	Sets the logging mode so that transaction information is written to a buffer before it is written to a logical log	None.
unbuf	Sets the logging mode so that data is not written to a buffer before it is written to a logical log	None.
nolog	Sets the logging mode so that no database transactions are logged	None.
ansi	Changes database logging to be ANSI compliant	**Additional Information:** Once you create or convert a database to ANSI mode, you cannot change it back to any of the other logging modes.
cancel	Cancels the logging-mode change request before the next level-0 backup occurs	None.
-f ***dbfile***	Changes the logging status of the databases that are listed (one per line) in the text file whose pathname is given by ***dbfile***	**Additional Information:** This command is useful if the list of databases is long or used often.
db_list	Names a space-delimited list of databases whose logging status is to be changed	**Additional Information:** If you do not specify anything, all databases that the database server manages are modified.

Chapter 8. The oninit Utility

oninit: Initialize the Database Server

Execute the **oninit** utility from the command line to initialize database server shared memory and bring the database server online. If you use the **oninit -i** option, you can also initialize disk space.

To initialize the database server in single-user mode, use the **oninit -j** option. This is an administrator-only mode you can use to perform maintenance operations including those that require executing SQL or DDL commands. You can use the **-j** flag with other **oninit** flags, except the **-s** flag. When in single-user mode, the system will only accept connection requests from the **informix** user. The server makes an entry in the online log whenever it enters or exits single-user mode.

On UNIX, you must be logged in as user **root** or **informix** to execute **oninit**. User informix should be the only member of the group **informix**. On Windows, you must be a member of the **Informix-Admin** group.

Before you initialize the database server, set the **INFORMIXSERVER** environment variable to the dbservername that you chose when you set the configuration parameter DBSERVERNAME. **INFORMIXSERVER** is not required for initialization. However, if **INFORMIXSERVER** is not set, the database server does not build the **sysmaster** tables. Also, the DB–Access utility requires **INFORMIXSERVER** to be set.

For information about what happens during initialization, see the chapter on initializing the database server in the *IBM Informix Administrator's Guide*.

Syntax

```
                                              (1)
►►──oninit──┬─┤ Initialize Shared Memory Only, ├──────────────┬──┬────┬──────►
            │                                        (2)      │  └─-y─┘
            └─┤ Initialize Disk Space and Shared Memory, ├────┘

►──┬────┬──────────────────────────────────────────────────────────────────►◄
   └─-j─┘
```

Notes:

1 See page 8-2

2 See page 8-3

Element	Purpose	Key Considerations
-y	Causes the database server to automatically respond yes to all prompts	None.
-j	Initializes the server in single-user mode	**Additional Information**: The **-j** flag may be combined with other **oninit** flags, except the quiescent mode (**-s**) flag.

Note: If you enabled the FAST_RESTART_PHYSLOG parameter by setting it to 1 and the database server shuts down, you can initiate fast recovery by executing **oninit** without any options.

Initialize Shared Memory Only

Initialize Shared Memory Only:

```
├──┬────┬──┬────┬──┬────┬─────────────────────────────────────────────┤
   └─-p─┘  └─-s─┘  └─-S─┘
```

Element	Purpose	Key Considerations
-p	Directs **oninit** not to search for (and delete) temporary tables	**Additional Information**: If you use this option, the database server returns to online mode more rapidly, but space used by temporary tables left on disk is not reclaimed.
-s	Initializes shared memory and leaves the database server in quiescent mode See "Initializing Shared Memory with the -s Option."	**Additional Information**: The database server should be in offline mode to initialize shared memory. **Additional Information**: Do not use this flag in combination with the **-j** flag. Specifying both **-j** and **-s** will result in an error.
-S	Starts database server in standard mode; disables HDR	**Additional Information**: If you use the **-S** option, the database server starts as a standard server instead of as a primary or as a secondary HDR server. It will leave the database server in quiescent mode and will require a subsequent **onmode -m** command for multiuser access.

Initializing Shared Memory with No Options

If you execute **oninit** without options, the database server is left in online mode after shared memory is initialized. For example, the following commands take the database server offline and back online:

```
onmode -ky
oninit
```

Initializing Shared Memory with the -s Option

The **-s** option initializes shared memory and leaves the database server in quiescent mode.

The following commands shut down and restart the database server in quiescent mode:

```
onmode -ky
oninit -s
```

Initialize Disk Space and Shared Memory

Initialize Disk Space and Shared Memory:

```
├──-i──┬────┬─────────────────────────────────────────────────────────────┤
       └─-s─┘
```

Element	Purpose	Key Considerations
-i	Causes the database server to initialize disk space and shared memory Leaves the database server in online mode after it initializes disk space	None.
-s	When used with **-i**, causes the database server to be left in quiescent mode after disk initialization	None.

When Dynamic Server 10.0 is first initialized with the **oninit -iyv** command, by default it comes online with large chunk mode fully enabled. Reversion is not possible. For more information about allowing large chunk mode, see “Allow Large Chunk Mode” on page 10-3

Warning: When you initialize disk space, the initialization destroys all data that your database server currently manages.

The database server must be offline when you initialize disk space.

Specify the Number of Virtual Processors

Use VPCLASS *cpu*,*num* and VPCLASS *aio* to specify the initial number of VPs for the CPU and AIO classes. For more information, see “VPCLASS” on page 1-85.

The VPCLASS configuration parameter allows you to specify, for each class of virtual processors, the number of VPs that the database server should start on initialization. Alternatively, you can use NUMCPUVPS and NUMAIOVPS to specify the initial number of VPs for the CPU and AIO classes. However, you cannot use both VPCLASS and NUMCPUVPS and NUMAIOVPS in the same configuration file. If your ONCONFIG file contains conflicting parameters, **oninit** returns one of the following messages:

```
oninit: Can't mix VPCLASS cpu and NUMCPUVPS, SINGLE_CPU_VP, AFF_SPROC, AFF_NPROCS, or NOAGE
        parameters

oninit: Can't mix VPCLASS aio and NUMAIOVPS parameters
```

For more information, refer to “VPCLASS” on page 1-85.

Chapter 9. The onlog Utility

onlog: Display Logical-Log Contents

The **onlog** utility displays the contents of a logical-log file, either on disk or on backup.

The **onlog** output is useful in debugging situations when you want to track a specific transaction or see what changes have been made to a specific tblspace. (For information about interpreting the logical-log file contents, see Chapter 4, “Interpreting Logical-Log Records,” on page 4-1)

Any user can run all of the **onlog** options except the **-l** option. Only user **informix** on UNIX or a member of the **Informix-Admin** group on Windows can run the **-l** option.

If the database server is in offline mode when you execute **onlog**, only the files on disk are read. If the database server is in quiescent or online mode, **onlog** also reads the logical-log records stored in the logical-log buffers in shared memory (after all records on disk have been read).

When the database server reads a logical-log file with status `U` from disk while in online mode, the database server denies all access to the logical-log files, effectively stopping database activity for all sessions. (For more information, see “onstat -l” on page 14-82.) For this reason, it is recommended that you wait until the files have been backed up and then read the contents of the logical-log files from backup.

Syntax

```
>>--onlog------------------------------------------------------------------>

                              (1)                                 (2)
>--| Log-Record Read Filters |-----| Log-Record Display Filters |-----+------+-><
                                                                       '--q--'
```

Notes:

1 See page 9-2

2 See page 9-3

Element	Purpose	Key Considerations
-q	Suppresses the initial header and the one-line header that appears every 18 records by default	None.

Read Filters

You direct **onlog** to read the following portions of the logical log as it searches for records to display:

- Records stored on disk
- Records stored on backup media
- Records from the specified logical-log file

By default, **onlog** displays the logical-log record header, which describes the transaction number and the record type. The record type identifies the type of operation performed.

In addition to the header, you can use the read filters to direct **onlog** to display the following information:

- Logical-log record header and data (including copies of simple large objects stored in a dbspace or tblspace)
- Copies of blobpages from blobspaces

  They are copied from the logical-log backup only. They are not available from disk.

Display Filters

You can display every logical-log record header, or you can specify output based on the following criteria:

- Records associated with a specific table
- Records initiated by a specific user
- Records associated with a specific transaction

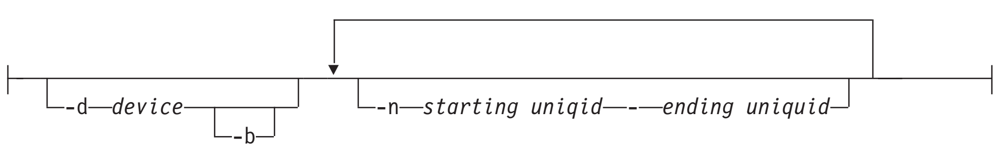
If an Error Is Detected

If **onlog** detects an error in the log file, such as an unrecognizable log type, it displays the entire log page in hexadecimal format and terminates.

Log-Record Read Filters

The **onlog** utility uses the pathnames that are stored in the root dbspace reserved pages to locate the logical-log files. If you use ON–Bar to back up the logical logs, **onlog** asks the storage manager to retrieve the appropriate logical-log records from the backup media.

Log-Record Read Filters:



Element	Purpose	Key Considerations
-b	Displays logical-log records associated with blobspace blobpages	**Additional Information**: The database server stores these records on the logical-log backup media as part of blobspace logging.
-d *device*	Names the pathname of the storage device where the desired logical-log backup is mounted	**Restriction:** If you use **ontape**, the device that you name must be the same as the pathname of the device assigned to the configuration parameter LTAPEDEV. If the **-d** option is not used, **onlog** reads the logical-log files stored on disk, starting with the logical-log file with the lowest *logid*. **Additional Information**: You do not need to use the **-d** option if you use ON–Bar because the storage manager retrieves the logical-log records from the storage device. **References**: For pathname syntax, see your operating-system documentation.
-n *starting uniqid-ending uniqid*	Directs **onlog** to read all the logical-log records contained in the log file that you specified from *starting uniqid* to the *ending uniqid*.	**Additional Information:** The *starting uniqid* and the *ending uniqid* are the unique ID numbers of the logical log. To determine the *uniqid* of a particular logical-log file, use the **onstat -l** command. If you do not use the **-n** option, **onlog** reads all the logical-log files that are available (either on disk or on tape). **References**: For information about the **onstat** utility, see “Monitor the Database Server Status” on page 14-3

Log-Record Display Filters

Log-Record Display Filters:

```
   .-----------------------------.
   V                             |
|----+-------------------------+-+--------------------------------------|
     |     (1)                 |
     +-l-----------------------+
     |                   (1)   |
     +-t--tblspace_num---------+
     |               (1)       |
     +-u--username-------------+
     |                     (1) |
     '-x--transaction_id-------'
```

Notes:

1 Only one occurrence of this item allowed

Element	Purpose	Key Considerations
-l	Displays the long listing of the logical-log record.	**Additional Information**: The long listing of a log record includes a complex hexadecimal and ASCII dump of the entire log record. The listing is not intended for casual use.

Element	Purpose	Key Considerations
-t ***tblspace_num***	Displays records associated with the tblspace that you specify.	**Restrictions**: Unsigned integer. Number, greater than 0, must be in the **partnum** column of the **systables** system catalog table. **Additional Information**: Specify this value as either an integer or hexadecimal value. (If you do not use a 0x prefix, the value is interpreted as an integer.) To determine the tblspace number of a particular tblspace, query the **systables** system catalog table as described in "Tblspace Numbers" on page 3-6.
-u ***username***	Displays records for a specific user.	**Restrictions**: User name must be an existing login name. User name must conform to operating-system-specific rules for login name.
-x ***transaction_id***	Displays only records associated with the transaction that you specify.	**Restriction**: Value must be an unsigned integer between 0 and TRANSACTIONS - 1, inclusive. **Additional Information**: You should need to use the **-x** option only in the unlikely case that an error is generated during a rollforward. When this situation occurs, the database server sends a message to the message log that includes the transaction ID of the offending transaction. You can use this transaction ID with the **-x** option of **onlog** to investigate the cause of the error.

If you do not specify any options, **onlog** displays a short listing of all the records in the log. You can combine options with any other options to produce more selective filters. For example, if you use both the **-u** and **-x** options, **onlog** displays only the activities that the specified user initiated during the specified transaction. If you use both the **-u** and **-t** options, **onlog** displays only the activities initiated by the specified user and associated with the specified tblspace.

Chapter 10. Changing Modes and Shared Memory with the onmode Utility

In This Chapter

The **onmode** flags determine which of the following operations **onmode** performs:

- Change the database server operating mode.
- Force a checkpoint.
- Control the B-tree scanner.
- Change residency of the resident and virtual portions of shared memory.
- Switch the logical-log file.
- Kill a database server session.
- Add a shared-memory segment to the virtual shared-memory portion.
- Add or remove virtual processors.
- Regenerate a **.infos** file.
- Set decision-support parameters.
- Free unused memory segments.

- Override the WAIT mode of the ONDBSPACEDOWN configuration parameter.
- Enable large chunks and chunk offsets to a maximum size of 4 terabytes and allow up to 32,766 total chunks.
- Revert data to an earlier database server format.

  For information about migrating from or reverting to earlier versions of the database server, see the *IBM Informix Migration Guide*.
- Set data-replication options.
- Replicate an index with Data-Replication.
- Set SQL statement cache options.
- Dynamically set the value of the SET EXPLAIN statement
- Dynamically update the value of certain connection, PDQ, and memory configuration parameters.

If you do not use any options, the database server returns a usage statement.

On UNIX, you must be user **root** or user **informix** to execute **onmode**. On Windows, you must be a member of the **Informix-Admin** group.

onmode Syntax

```
>>--onmode--+------------------------------------------------------------------+--+----+----><
            |                                  (1)                             |  '-y-'
            +-| Allow Large Chunks Mode |--------------------------------------+
            |                                  (2)                             |
            +-| Change Database Server Mode |----------------------------------+
            |                                  (3)                             |
            +-| Force a Checkpoint |-------------------------------------------+
            |                                  (4)                             |
            +-| Switch the Logical-Log File |----------------------------------+
            |                                  (5)                             |
            +-| Kill a Database Server Session |-------------------------------+
            |                                  (6)                             |
            +-| Add a Shared-Memory Segment |----------------------------------+
            |                                  (7)                             |
            +-| Add or Remove Virtual Processors |-----------------------------+
            |                                  (8)                             |
            +-| Regenerate .infos File |---------------------------------------+
            |                                  (9)                             |
            +-| Change Decision-Support Parameters |---------------------------+
            |                                  (10)                            |
            +-| Free Unused Memory Segments |----------------------------------+
            |                                  (11)                            |
            +-| Override ONDBSPACEDOWN WAIT Mode |-----------------------------+
            |                                  (12)                            |
            +-| Change Shared-Memory Residency |-------------------------------+
            |                                  (13)                            |
            +-| Kill a Distributed Transaction |-------------------------------+
            |                                  (14)                            |
            +-| Set Data-Replication Types |-----------------------------------+
            |                                  (15)                            |
            +-| Replicate an Index with Data-Replication |---------------------+
            |                                  (16)                            |
            +-| Change Usage of the SQL Statement Cache |----------------------+
            |                                  (17)                            |
            +-| Change Settings for the SQL Statement Cache |------------------+
            |                                  (18)                            |
            +-| Dynamically Setting of SET EXPLAIN |---------------------------+
            |                                  (19)                            |
            +-| Dynamically Change Certain Connection, PDQ, and Memory Parameters |-+
            |                                  (20)                            |
            '-| Change Database Format with onmode -b |------------------------'
```

Notes:

1 See page 10-3

2 See page 10-4

3 See page 10-6

4 See page 10-8

5 See page 10-8

6 See page 10-11

7 See page 10-12

8 See page 10-14

9 See page 10-15

10 See page 10-16

11 See page 10-17

12 See page 10-8

13 See page 10-9

14 See page 10-9

15 See page 10-10

16 See page 10-17

17 See page 10-18

18 See page 10-19

19 See page 10-19

20 See the *IBM Informix Migration Guide*

Element	Purpose	Key Considerations
-y	Causes the database server to automatically respond yes to all prompts	None.

Allow Large Chunk Mode

Allow Large Chunks Mode:

```
├──┬─-BC 1─┬───────────────────────────────────────────────────┤
   └─-BC 2─┘
```

Element	Purpose	Key Considerations
-BC 1	Enables support of large chunks, large offsets that are greater than 2 GB, and allows more than 2047 chunks per dbspace.	**Additional Information**: This option allows large chunks to be created. Reversion is possible if there are no large chunk features in the server, the root chunk or none added. Dbspaces and blobspaces without chunks greater than 2 GB remain in the old version. After a chunk larger than 2 GB is added to a dbspace or blobspace then all chunks added or altered in that dbspace or blobspace are in the new format. **References:** See your *IBM Informix Administrator's Guide*.

Element	Purpose	Key Considerations
-BC 2	Allows large-chunk-only mode for all dbspaces.	**Additional Information:** Reversion is not possible. Enables the 9.4 large chunk feature for all dbspaces and blobspaces, Any chunk or offset added or modified has the new format. Existing chunks that you do not alter remain in the old format. **References**: See your *IBM Informix Administrator's Guide*

The **onmode -BC** (backward-compatible) commands are useful if you have converted from Dynamic Server 9.40 (small chunk mode) to Dynamic Server 10.0. When Dynamic Server 10.0 is first initialized (with the **oninit -iyv** command), by default it comes online with large chunk mode already fully enabled. Reversion is not possible. In the case of a newly initialized instance of Dynamic Server 10.0, the **onmode -BC** commands will return an error.

Note: After executing the **onmode -BC** command, perform a complete system level-0 backup.

Change Database Server Mode

Change Database Server Mode:

```
├──┬─-k─┬──────────────────────────────────────────────────────────────┤
   ├─-m─┤
   ├─-s─┤
   ├─-u─┤
   └─-j─┘
```

Element	Purpose	Key Considerations
-k	Takes the database server to offline mode and removes shared memory	**Additional Information**: To reinitialize shared memory, shut down and restart the database server. **References:** See "Taking the Database Server to Offline Mode with the -k Option" on page 10-5.
-m	Takes the database server from quiescent or single-user mode to online mode	**References**: See "Bringing the Database Server Online with the -m Option" on page 10-5.
-s	Shuts down the database server gracefully	**Additional Information**: Users who are using the database server are allowed to finish before the database server comes to quiescent mode, but no new connections are allowed. When all processing is finished, **-s** takes the database server to quiescent mode. The **-s** option leaves shared memory intact. **References**: See "Shutting Down the Database Server Gracefully with the -s Option" on page 10-5.
-u	Shuts down the database server immediately	**Additional Information**: This option brings the database server to quiescent mode without waiting for users to finish their sessions. Their current transactions are rolled back, and their sessions are terminated. **References**: See "Shutting Down the Database Server Immediately with the -u Option" on page 10-5.

Element	Purpose	Key Considerations
-j	Puts the database server into single-user mode	**Additional Information**: This option brings the database server to single-user mode, allowing the **informix** user all functions including the issuance of SQL and DDL commands. **References:** See your *IBM Informix Administrator's Guide*.

The following sections describe the options that take the database server from one mode to another.

Taking the Database Server to Offline Mode with the -k Option

The **-k** option takes the database server to offline mode and removes database server shared memory.

A prompt asks for confirmation. Another prompt asks for confirmation to kill user threads before the database server comes offline. If you want to eliminate these prompts, execute the **-y** option with the **-s** option.

This option does not kill all client sessions. Use the **-u** option to avoid hanging client sessions or virtual server processes.

Important: When you use the **onmode -k** command to shut down the database server, utilities that are waiting for a user response might not terminate. For example, **ontape** might be waiting for another tape, **onstat -i** might be waiting for a user response, or **onspaces** might be waiting for **y** or **n** to continue. If this problem occurs, use **onmode -uk** or **-uky** instead to roll back work before removing shared memory. For more information, see the descriptions of other options on this page.

Bringing the Database Server Online with the -m Option

The **-m** option brings the database server online from quiescent mode.

Shutting Down the Database Server Gracefully with the -s Option

The **-s** option causes a graceful shutdown. Users who are using the database server are allowed to finish before the database server comes to quiescent mode, but no new connections are allowed. When all processing is finished, **-s** takes the database server to quiescent mode. The **-s** option leaves shared memory intact.

A prompt asks for confirmation. If you want to eliminate this prompt, execute the **-y** option with the **-s** option.

Shutting Down the Database Server Immediately with the -u Option

The **-u** option causes immediate shutdown. This option brings the database server to quiescent mode without waiting for users to finish their sessions. Their current transactions are rolled back, and their sessions are terminated.

A prompt asks for confirmation. Another prompt asks for confirmation to kill user threads before the database server comes to quiescent mode. If you want to eliminate these prompts, execute the **-y** option with the **-s** option.

Changing the Mode of the Database Server to Single-user with the -j Option

The **-j** option puts the database server into a maintenance mode in which only the **informix** user is allowed to connect to the server. Once connected, the **informix** user can execute any SQL or DDL commands. This allows a DBA to have the server in a fully functional mode to perform maintenance.

Changing Database Server Mode with ON-Monitor (UNIX)

You can also use ON–Monitor options to change the database server mode. The following table shows ON–Monitor options that are equivalent to the **onmode** options.

onmode Option	ON-Monitor Option
-k	Take-Offline
-m	On-Line
-s	Graceful-Shutdown
-u	Immediate-Shutdown
-j	Single-user Mode

Force a Checkpoint

Force a Checkpoint:

```
├──-c──┬─────────┬──────────────────────────────────────────────┤
       ├─fuzzy───┤
       ├─block───┤
       └─unblock─┘
```

Element	Purpose	Key Considerations
-c	Forces a checkpoint that flushes the buffers to disk	**Additional Information**: You can use the **-c** option to force a sync checkpoint if the most recent checkpoint record in the logical log was preventing the logical-log file from being freed (status U-B-L).
block	Blocks the database server from any transactions	**Additional Information**: While the database server is blocked, users can access it in read-only mode. Use this option to perform an external backup on Dynamic Server. **References:** For more information, see the *IBM Informix Backup and Restore Guide*.
unblock	Unblocks the database server	When the database server is unblocked, data transactions and normal database server operations can resume. Use this option after you complete an external backup on Dynamic Server. **References:** For more information, see the *IBM Informix Backup and Restore Guide*.

Element	Purpose	Key Considerations
fuzzy	Performs a fuzzy checkpoint	**Additional Information**: Use the **onmode -c fuzzy** option to force a fuzzy checkpoint. Then use the **onstat -b** command to check the number of modified (dirty) buffers. The modified buffers that are still in the buffer cache contain fuzzy transactions. **References:** For more information, see the chapter on checkpoints and fast recovery in the *IBM Informix Administrator's Guide*.

Control the B-tree Scanner

Control B-tree scanner:

```
├──-C──┬────────────────┬──────────────────────────────────────────────┤
       ├─start──────────┤
       ├─stop──count────┤
       ├─kill──count────┤
       ├─threshold──size┤
       ├─high───────────┤
       └─low────────────┘
```

Element	Purpose	Key Considerations
-C	Controls the B-tree scanner for cleaning indexes of deleted items	**Additional Information**: There is no limit to the number of threads that can run at one time. However, there is a limit of 128 threads that can be started at one time. If, for example, you wanted 150 threads to run, you could execute two commands: **onmode -C 100** and **onmode -C 50.**
stop *count* **kill** *count*	Stops or kills the B-tree scanner threads	**Additional Information**: Either of these threads stops or kills the B-tree scanners. If a count you specify is higher than the number of threads currently running, the current number is assumed. If you do not specify a number, then a count of 1 is assumed.
threshold *size*	Sets the minimum number of deleted items an index must encounter before placing a priority or hot list	**Additional Information**: Once all indexes above the threshold are cleaned, then indexes below the threshold are added to the hot list. The default threshold is 500. **onmode -C threshold -1** cleans every index in the database. **onmode -C threshold 0** cleans every index that has a dirty_hit count greater than 0.
high	Sets the priority of all B-tree scanner threads that are running	**Additional Information**: This option sets the priority of the B-tree scanner threads to equal that of normal users.
low	Sets the priority of all B-tree scanner threads that are running	**Additional Information**: This option sets the priority of the B-tree scanner threads lower than that of normal users. This command allows the B-tree scanner to consume only spare system resources, ensuring that the threads will not use the CPU cycles of normal users. The default priority is low.

The B-tree scanner assigns a profile to each index, depending on the amount of extra work the index places on the server. From the index profiles, the B-tree scanner develops a hot list: `btc_create_hot_list`. The B-tree scanner keeps track of the number of times items in an index causes the server to do extra work and cleans that index first. The index causing the next highest amount of extra work will then be cleaned, and so on in descending order. The B-tree scanner allocates cleaning threads dynamically, which allows for configurable workloads.

Change Shared-Memory Residency

Change Shared- Memory Residency:

```
├──┬─-n─┬──────────────────────────────────────────────────────┤
   └─-r─┘
```

Element	Purpose	Key Considerations
-n	Ends forced residency of the resident portion of shared memory	**Additional Information**: This command does not affect the value of RESIDENT, the forced-residency parameter in the ONCONFIG file.
-r	Starts forced residency of the resident portion of shared memory	**Additional Information**: This command does not affect the value of RESIDENT, the forced-memory parameter in the ONCONFIG file.

Important: Set the RESIDENT parameter to 1 before you use the **onmode -r** or **-n** options.

For information on using the forced-residency parameter to turn residency on or off for the next time that you restart the database server, see the chapter on managing shared memory in the *IBM Informix Administrator's Guide*.

Switch the Logical-Log File

Switch the Logical- Log File:

```
├──-l──────────────────────────────────────────────────────────┤
```

Element	Purpose	Key Considerations
-l	Switches the current logical-log file to the next logical-log file	**Additional Information**: You must use **onmode** to switch to the next logical-log file. **References**: For information on switching to the next logical-log file, see the chapter on managing logical-log files in the *IBM Informix Administrator's Guide*.

Kill a Database Server Session

Kill a Database Server Session:

```
├──-z──sid─────────────────────────────────────────────────────┤
```

Element	Purpose	Key Considerations
-z ***sid***	Kills the session that you specify in *sid*	**Restrictions**: This value must be an unsigned integer greater than 0 and must be the session identification number of a currently running session.

To use the **-z** option, first obtain the session identification (*sessid*) with **onstat -u**, then execute **onmode -z**, substituting the session identification number for *sid*.

When you use **onmode -z**, the database server attempts to kill the specified session. If the database server is successful, it frees any resources that the session holds. If the database server cannot free the resources, it does not kill the session.

If the session does not exit the section or release the latch, the database server administrator can take the database server offline, as described in "Taking the Database Server to Offline Mode with the -k Option" on page 10-5, to close all sessions.

Kill a Distributed Transaction

Kill a Distributed Transaction:

```
├──-Z──address──────────────────────────────────────────────────┤
```

Element	Purpose	Key Considerations
-Z address	Kills a distributed transaction associated with the shared-memory address *address*	**Restrictions**: This argument must be the address of an ongoing distributed transaction that has exceeded the amount of time that TXTIMEOUT specifies. The address must conform to the operating-system-specific rules for addressing shared-memory. (The address is available from **onstat -x** output.) **Additional Information**: This option is not valid until the amount of time that the ONCONFIG parameter TXTIMEOUT specifies has been exceeded. The **-Z** option should rarely be used and only by an administrator of a database server involved in distributed transactions. **References**: For information on initiating independent actions in a two-phase commit protocol, see the chapter on multiphase commit protocols in the *IBM Informix Administrator's Guide*.

Distributed transactions provide the ability to query data on different database servers.

Warning: If applications are performing distributed transactions, killing one of the distributed transactions can leave your client/server database system in an inconsistent state. Try to avoid this situation.

Set Data-Replication Types

Set Data-Replication Types:

```
├──-d──┬─standard─────────────────────┬─────────────────────────┤
       └─┬─primary───┬──dbservername──┘
         └─secondary─┘
```

Element	Purpose	Key Considerations
-d	Used to set the High-Availability Data-Replication type, either standard, primary, or secondary, as the following sections describe	**Restrictions**: You can use the **-d primary** and **-d secondary** options only when the database server is in quiescent mode. You can use the **-d standard** option when the database server is in quiescent, online, or read-only mode.
dbservername	Identifies the database server name of the primary or secondary database server	**Restrictions**: The *dbservername* argument must correspond to the DBSERVERNAME parameter in the ONCONFIG file of the intended secondary database server. It should *not* correspond to one of the database servers that the DBSERVERALIASES parameter specifies. **Additional Information**: The *dbservername* argument of the other database server in the data-replication pair and the type of a database server (standard, primary, or secondary) is preserved after reinitialization of shared memory. **References**: For more information, see *range of values* for the DBSERVERNAME configuration parameter in "DBSERVERNAME" on page 1-22.

Using the -d standard Option

The **-d standard** option drops the connection between database servers in a data replication pair (if one exists) and sets the database server type of the current database server to standard. This option does not change the mode or type of the other database server in the pair.

Using the -d primary dbservername Option

The **-d primary** ***dbservername*** option sets the database server type to primary and attempts to connect with the database server that ***dbservername*** specifies. If the connection is successful, data replication is turned on. The primary database server goes into online mode, and the secondary database server goes into read-only mode. If the connection is not successful, the database server comes to on-line mode, but data replication is not turned on.

Using the -d secondary dbservername Option

The **-d secondary** ***dbservername*** option sets the database server type to secondary and attempts to connect with the database server that ***dbservername*** specifies. If the connection is successful, data replication is turned on. The primary database server goes online, and the secondary database server goes into read-only mode. If the connection is not successful, the database server comes to read-only mode, but data replication is not turned on.

Replicate an Index with Data-Replication

Replicate an Index:

```
├──-d──┬─idxauto──┬─on──┬──────────────────────────┬──────────────────────────────┤
       │          └─off─┘                          │
       └─index──database:──┬────────┬─table#index──┘
                           └─owner.─┘
```

Element	Purpose	Key Considerations
-d	Specifies how indexes are replicated to a High-Availability Data-Replication (HDR) secondary server when an index on the secondary server becomes corrupt	You can use the **onmode -d idxauto** and **-d index** commands while the server is in online mode.
idxauto	Enables automatic index replication when an index on a secondary server becomes corrupt	Use **onmode -d idxauto** to overwrite the value of the DRIDXAUTO configuration parameter within a session. **References:** For more information on DRIDXAUTO, see "DRIDXAUTO" on page 1-28. For more information on replicating indexes, see the chapter on using HDR in the *IBM Informix Administrator's Guide*.
index	Replicates an index from a primary to a secondary server	If you detect a corrupt index on a secondary server, use the **onmode -d index** command to start replication of the index from the primary to the secondary server.
database	Specifies the database containing the index to replicate	**References**: Syntax must conform to the Identifier segment; see *IBM Informix Guide to SQL: Syntax.*
index	Specifies the name of the index to replicate	**Restrictions**: Index must exist on table and in database specified. **References**: Syntax must conform to the Identifier segment; see *IBM Informix Guide to SQL: Syntax.*
owner	Specifies the owner of a table	**Restrictions**: You must specify the current owner of the table. **References**: Syntax must conform to the Table Name segment; see *IBM Informix Guide to SQL: Syntax.*
table	Specifies the name of the table on which the index is based	**References**: Syntax must conform to the Table Name segment; see *IBM Informix Guide to SQL: Syntax.*

The **-d idxauto** and the **-d index** options provide methods to replicate an index to a secondary server containing a corrupted index. The base table will be locked during the transfer of an index. The alternative to using these options is to drop and rebuild the corrupt index on the primary server.

In the case of a fragmented index with one corrupt fragment, the **-d idxauto** option only transfers the single affected fragment, whereas the **-d index** option transfers the whole index.

Add a Shared-Memory Segment

Add a Shared-Memory Segment:

```
├──-a──seg_size──────────────────────────────────────────────────┤
```

Element	Purpose	Key Considerations
-a *seg_size*	Allows you to add a new virtual shared-memory segment. Size is specified in kilobytes	**Restrictions**: The value of ***seg_size*** must be a positive integer. It must not exceed the operating-system limit on the size of shared-memory segments.

Ordinarily, you do not need to add segments to the virtual portion of shared memory because the database server automatically adds segments as they are needed. However, as segments are added, the database server might reach the operating-system limit for the maximum number of segments before it acquires the memory that it needs. This situation typically occurs when SHMADD is set so small that the database server exhausts the number of available segments before it acquires the memory that it needs for some operation.

If you manually add a segment that is larger than the segment specified by SHMADD, you can avoid exhausting the operating-system limit for segments but still meet the need that the database server has for additional memory.

Add or Remove Virtual Processors

Add or Remove Virtual Processors:

```
├──-p──┬──┬─────┬──number──┬─AIO──────┬─┬──────────────────────┤
       │  └─ + ─┘          ├─CPU──────┤ │
       │                   ├─ENCRYPT──┤ │
       │                   ├─JVP──────┤ │
       │                   ├─LIO──────┤ │
       │                   ├─PIO──────┤ │
       │                   ├─SHM──────┤ │
       │                   ├─SOC──────┤ │
       │                   ├─TLI──────┤ │
       │                   └─vpclass──┘ │
       └─ -──number──┬─CPU──────┬───────┘
                     ├─ENCRYPT──┤
                     ├─JVP──────┤
                     └─vpclass──┘
```

Element	Purpose	Key Considerations
-p *number*	Adds or removes virtual processors. The *number* argument indicates the number of virtual processors to add or remove If this value is a negative integer, processors are removed. If this value is a positive integer, processors are added.	**Restrictions**: You can use the **-p** option only when the database server is in online mode, and you can add to only one class of virtual processors at a time. For more details, see "Adding and Dropping Virtual Processors" on page 10-13. **Limits**: If you are removing virtual processors, the maximum cannot exceed the actual number of processors of the specified type. If you are adding virtual processors, the maximum number depends on the operating system. **References:** For more information, see the chapter on using virtual processors in the *IBM Informix Administrator's Guide*.
AIO	Performs nonlogging disk I/O to cooked disk spaces	Also performs nonlogging I/O to raw disk spaces if kernel asynchronous I/O (KAIO) is not used.

Element	Purpose	Key Considerations
CPU	Runs all session threads and some system threads	**Limits:** It is recommended that the number of CPU VPs not be greater than the number of physical processors. If KAIO is used, performs I/O to raw disk spaces, including I/O to physical and logical logs. Runs thread for KAIO where available or a single poll thread. The database server uses the number of CPU VPs to allocate resources for parallel database queries (PDQ). If you drop CPU VPs, your queries will run significantly slower. The **Reinit** field of the **onstat -g mgm** output displays information on the number of queries that are waiting for running queries to complete after an **onmode -p** command. Also see the *IBM Informix Performance Guide*.
ENCRYPT	Executes column-level encryption and decryption routines	Specify more ENCRYPT virtual processors if you have multiple encrypted columns.
JVP	Executes Java user-defined routines in the Java Virtual Machine (JVM)	Specify more JVPs if you are running many Java UDRs.
LIO	Writes to the logical-log files if they are in cooked disk space	Use two LIO virtual processors only if the logical logs are in mirrored dbspaces. The database server allows a maximum of two LIO virtual processors.
PIO	Writes to the physical log if it is in cooked disk space	Use two PIO virtual processors only if the physical log is in a mirrored dbspace. The database server allows a maximum of two PIO virtual processors.
SHM	Performs shared-memory communication	You can use the SHM virtual processor even if the database server is not configured for shared-memory communication.
SOC	Uses sockets to perform network communications	You can use the SOC virtual processor only if the database server is configured for network connections through sockets.
STR	Performs stream pipe connections	
TLI	Uses the Transport Layer Interface (TLI) to perform network communication	You can use the TLI virtual processor only if the database server is configured for network connections through TLI.
vpclass	Names a user-defined virtual processor class	**Additional Information**: Use the VPCLASS parameter in the ONCONFIG to define the user-defined virtual-processor class. Specify more user-defined virtual processors if you are running many UDRs. **Restrictions:** On Windows, you can have only one user-defined virtual processor class at a time. Omit the *number* parameter in the **onmode -p** *vpclass* command. **References:** For more information on extension classes, see “VPCLASS” on page 1-85.

Adding and Dropping Virtual Processors

The following rules about adding or dropping virtual processors apply:

- You can add but not drop virtual processors of the AIO, PIO, LIO, TLI, SHM, SOC, and STR classes.
- You cannot add or drop virtual processors of the OPT, ADM, ADT, and MSC classes. The database server adds them automatically.
- You can add or drop virtual processors of the CPU, ENCRYPT, JVP, and user-defined (*vpclass*) classes.

Windows Only

- On Windows, you can add a virtual processor of any class, but you cannot drop virtual processors.

End of Windows Only

Dropping Virtual Processors Automatically

Table 10-1 shows the virtual processors that the database server starts automatically. You cannot add or drop these virtual processors with the **onmode -p** command. To drop these virtual processors, shut down and restart the database server.

Table 10-1. Virtual-Processor Classes That the Database Server Starts Automatically

Virtual-Processor Class	Description
ADM	Performs administrative functions
ADT	Runs auditing processes The database server starts one virtual processor in the audit class when you turn on audit mode by setting the ADTMODE parameter in the ONCONFIG file.
MSC	Services requests for system calls that require a large stack The database server starts this virtual processor automatically.
OPT	Performs I/O to the optical disk The database server starts one OPT virtual processor when you use the Optical Subsystem.

Monitoring Poll Threads with onstat

While the database server is online, you cannot drop a CPU virtual processor that is running a poll thread. To identify poll threads that run on CPU virtual processors, use the following command:

```
onstat -g ath | grep 'cpu.*poll'
```

The following **onstat -g ath** output shows two CPU virtual processors with poll threads. In this situation, you cannot drop to fewer than two CPU virtual processors.

```
tid  tcb      rstcb  prty  status    vp-class  name

8     a362b90  0     2     running   1cpu      tlitcppoll
9     a36e8e0  0     2     cond wait arrived   3cpu
```

For more information on the types of virtual processors, see the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide*.

Regenerate .infos File

Regenerate .infos File:

```
├──-R────────────────────────────────────────────────────────────────────────┤
```

Element	Purpose	Key Considerations
-R	Re-creates the **.infos.dbservername** file	**Restrictions**: Before you use the **-R** option, set the **INFORMIXSERVER** environment variable to match the DBSERVERNAME parameter from the ONCONFIG file. Do not use the **-R** option if **INFORMIXSERVER** is one of the DBSERVERALIAS names. **Additional Information:** For more information, see ".infos.dbservername" on page A-6.

The database server uses information from the **.infos.dbservername** file when it accesses utilities. The database server creates and manages this file, and you should never need to do anything to the file. However, if **.infos.dbservername** is accidentally deleted, you must either recreate the file or shut down and restart the database server.

Change Decision-Support Parameters

Change Decision- Support Parameters:

```
├──┬─-D──max_priority─┬──────────────────────────────────────────────┤
   ├─-M──kilobytes────┤
   ├─-Q──queries──────┤
   └─-S──scans────────┘
```

Element	Purpose	Key Considerations
-D ***max_priority***	Changes the value of MAX_PDQPRIORITY	**Restrictions**: This value must be an unsigned integer between 0 and 100. **Additional Information**: Specify *max_priority* as a factor to temper user requests for PDQ resources. **References**: For information on parameters used for controlling PDQ, see "MAX_PDQPRIORITY" on page 1-50and the *IBM Informix Performance Guide.*
-M ***kilobytes***	Changes the value of DS_TOTAL_MEMORY	**Restrictions**: This value has a platform-dependent upper limit. The value for 32-bit systems must be an unsigned integer between 128 * DS_MAX_QUERIES and 1,048,576. On 64-bit systems, the limit is generally higher and varies with the operating system. On HP 9000 platforms, for example, the maximum value is 4,294,967,296. **Additional Information**: Specify *kilobytes* for the maximum amount of memory available for parallel queries. **References**: For more information, see "DS_TOTAL_MEMORY" on page 1-34 and the *IBM Informix Performance Guide*.
-Q ***queries***	Changes the value of DS_MAX_QUERIES	**Restrictions**: This value must be an unsigned integer between 1 and 8,388,608. **Additional Information**: Specify *queries* for the maximum number of concurrently executing parallel queries. **References**: For information on parameters used for controlling PDQ, see "DS_MAX_QUERIES" on page 1-31and the *IBM Informix Performance Guide.*

Element	Purpose	Key Considerations
-S *scans*	Changes the value of DS_MAX_SCANS	**Restrictions**: This value must be an unsigned integer between 10 and 1,048,576. **Additional Information**: Specify *scans* for the maximum number of concurrently executing parallel scans. **References**: For information on parameters used for controlling PDQ, see "DS_MAX_SCANS" on page 1-32and the *IBM Informix Performance Guide*.

These options allow you to change configuration parameters while the database server is online. The new values affect only the current instance of the database server; the values are not recorded in the ONCONFIG file. If you shut down and restart the database server, the values of the parameters revert to the values in the ONCONFIG file. For more information about these configuration parameters, see Chapter 1, "Configuration Parameters," on page 1-1.

To check the current values for the MAX_PDQPRIORITY, DS_TOTAL_MEMORY, DS_MAX_SCANS, DS_MAX_QUERIES, and the DS_NONPDQ_QUERY_MEM configuration parameters, use **onstat -g mgm**. See "The onstat -g mgm Option" on page 14-47.

Free Unused Memory Segments

Free Unused Memory Segments:

```
├──-F──────────────────────────────────────────────────────────┤
```

Element	Purpose	Key Considerations
-F	Frees unused memory segments	None.

When you execute **onmode -F**, the memory manager examines each memory pool for unused memory. When the memory manager locates blocks of unused memory, it immediately frees the memory. After the memory manager checks each memory pool, it begins checking memory segments and frees any that the database server no longer needs.

It is recommended that you run **onmode -F** from an operating-system scheduling facility regularly and after the database server performs any function that creates additional memory segments, including large index builds, sorts, or backups.

Running **onmode -F** causes a significant degradation of performance for any users that are active when you execute the utility. Although the execution time is brief (1 to 2 seconds), degradation for a single-user database server can reach 100 percent. Systems with multiple CPU virtual processors experience proportionately less degradation.

To confirm that **onmode** freed unused memory, check your message log. If the memory manager frees one or more segments, it displays a message that indicates how many segments and bytes of memory were freed.

Override ONDBSPACEDOWN WAIT Mode

Override ONDBSPACEDOWN WAIT Mode:

```
├──-O──────────────────────────────────────────────────────────┤
```

Element	Purpose	Key Considerations
-O	Overrides the WAIT mode of the ONDBSPACEDOWN configuration parameter	None.

Use the **onmode -O** option only in the following circumstances:

- ONDBSPACEDOWN is set to WAIT.
- A disabling I/O error occurs that causes the database server to block all updating threads.
- You cannot or do not want to correct the problem that caused the disabling I/O error.
- You want the database server to mark the disabled dbspace as down and continue processing.

When you execute this option, the database server marks the dbspace responsible for the disabling I/O error as down, completes a checkpoint, and releases blocked threads. Then **onmode** prompts you with the following message:

```
This will render any dbspaces which have incurred disabling I/O errors unusable
and require them to be restored from an archive.
Do you wish to continue?(y/n)
```

If **onmode** does not find any disabling I/O errors on noncritical dbspaces when you run the -O option, it notifies you with the following message:

```
There have been no disabling I/O errors on any noncritical dbspaces.
```

Change Usage of the SQL Statement Cache

Change Usage of the SQL Statement Cache:

```
├──-e──mode────────────────────────────────────────────────────┤
```

Element	Purpose	Key Considerations
onmode -e `ENABLE`	Enables the SQL statement cacheFor more information, see the material on improving query performance in the *IBM Informix Performance Guide*.	User sessions use the cache only when they perform either of the following actions: • Set the environment variable **STMT_CACHE** to 1 • Execute the SQL statement SET STATEMENT CACHE ON
onmode -e `FLUSH`	Flushes the statements that are not in use from the SQL statement cache	The **onstat -g ssc ref_cnt** field shows 0.
onmode -e `OFF`	Turns off the SQL statement cache	No statements are cached.

Element	Purpose	Key Considerations
onmode -e ON	Turns on the SQL statement cache	All statements are cached unless the user turns it off with one of the following actions: • Set the environment variable **STMT_CACHE** to 0 • Execute the SQL statement SET STATEMENT CACHE OFF

The **onmode -e** changes are in effect for the current database server session only. When you restart the database server, it uses the default STMT_CACHE parameter value in the **ONCONFIG** file.

Change Settings for the SQL Statement Cache

Change Settings for the SQL Statement Cache:

```
├──-W──┬─STMT_CACHE_HITS──hits──────┬────────────────────────────────────────────┤
       └─STMT_CACHE_NOLIMIT──value──┘
```

Element	Purpose	Key Considerations
STMT_CACHE_HITS *hits*	Specifies the number of hits (references) to a statement before it is fully inserted in the SQL statement cache Set ***hits*** to 1 or more to exclude ad hoc queries from entering the cache.	You can only increase or reset the value of STMT_CACHE_HITS. The new value displays in the **#hits** field of the **onstat-g ssc** output. If ***hits*** = 0, the database server inserts all qualified statements and its memory structures in the cache. If ***hits*** > 0 and the number of times the SQL statement has been executed is less than STMT_CACHE_HITS, the database server inserts *key-only* entries in the cache. It inserts qualified statements in the cache after the specified number of hits have been made to the statement. **ONCONFIG** Parameter: STMT_CACHE_HITS
STMT_CACHE_NOLIMIT *value*	Controls whether statements are inserted in the SQL statement cache when its size is greater than the STMT_CACHE_SIZE value	If *value* = 0, the database server inserts statements in the cache when its size is greater than the value of STMT_CACHE_SIZE. If *value* = 1, the database server always inserts statements in the cache. If none of the queries are shared, turn off STMT_CACHE_NOLIMIT to prevent the database server from allocating a large amount of memory for the cache.**ONCONFIG** Parameter: STMT_CACHE_NOLIMIT

SQL Statement Cache Examples

The following are examples of **onmode -W** commands for changing SQL statement cache (SSC) settings. The changes are in effect for the current database server session only and do not change the ONCONFIG values. When you restart the database server, it uses the default SSC settings, if not specified in the ONCONFIG file, or the ONCONFIG settings. To make the changes permanent, set the appropriate configuration parameter.

```
onmode -W STMT_CACHE_HITS 2  # number of hits before statement is
     # inserted into SSC
onmode -W STMT_CACHE_NOLIMIT 1  # always insert statements into
     # the cache
```

Dynamically Setting of SET EXPLAIN

Dynamically Change SQEXPLAIN Setting:

```
├──-Y──sessionid──┬─1─┬──────────────────────────────────────────────┤
                  └─0─┘
```

Element	Purpose	Key Considerations
-Y	Dynamically set value of the SET EXPLAIN statement.	None.

You can use the SET EXPLAIN statement to display the query plan of the optimizer, an estimate of the number of rows returned, and the relative cost of the query. When you use the **onmode -Y** command to turn on SET EXPLAIN, the output is displayed in the **sqexplain.out.*sessionid*** file. If an **sqexplain.out** file already exists, the database server uses that file until an administrator turns off the dynamic explain for the session.

The **onmode -Y** command dynamically changes the value of the SET EXPLAIN statement for an individual session. The following invocations are valid with this command:

Invocation	Explanation
onmode -Y *sessionid* 1	Turns dynamic explain on for ***sessionid***
onmode -Y *sessionid* 0	Turns dynamic explain off for ***sessionid***

For more information on using the SET EXPLAIN statement, see the *IBM Informix Guide to SQL: Syntax*. For more information on interpreting the **sqexplain.out** file to improve query performance, see the *IBM Informix Performance Guide*.

Dynamically Change Certain Connection, PDQ, and Memory Parameters

Dynamically Change Certain Connection, PDQ, and Memory Parameters:

```
├──┬─-wm──config_param=value─┬──────────────────────────────────────┤
   └─-wf──config_param=value─┘
```

Element	Purpose	Key Considerations
-wm	Dynamically set value of the specified configuration parameter for the current session.	None.
-wf	Update the value of the specified configuration parameter in the ONCONFIG file.	None.

Element	Purpose	Key Considerations
config_param=*value*	The configuration parameter and its new value. The following configuration parameters can be specified: • DS_MAX_QUERIES • DS_MAX_SCANS • DS_NONPDQ_QUERY_MEM • DS_TOTAL_MEMORY • LISTEN_TIMEOUT • ONLIDX_MAXMEM • MAX_INCOMPLETE_CONNECTIONS • MAX_PDQPRIORITY • RESIDENT	**References:** Chapter 1, "Configuration Parameters," on page 1-1.

Chapter 11. The ON-Monitor Utility

Using ON-Monitor (UNIX)

Use the ON–Monitor utility to perform various administrative tasks. This section provides a quick reference for the ON–Monitor screens. To start ON–Monitor, execute the following command from the operating-system prompt:

```
onmonitor
```

If you are logged in as user **informix** or user **root**, the main menu appears. All users other than **informix** and **root** have access only to the Status menu.

The ON–Monitor main menu displays the following menus:

- **Status** menu
- **Parameters** menu
- **Dbspaces** menu
- **Mode** menu
- **Force-Ckpt** menu
- **Archive** menu
- **Logical-Logs** menu
- **Exit** option

These menus are shown on the following pages (Table 11-1 on page 11-2 through Table 11-7 on page 11-3).

Navigating ON-Monitor and Using Help

All menus and screens in ON–Monitor function in the same way. For menus, use the arrow keys or `SPACEBAR` to scroll to the option that you want to execute and press `RETURN`, or press the first capitalized letter of the option (usually the first letter). When you move from one option to the next by pressing `SPACEBAR` or an arrow key, the option explanation (line 2 of the menu) changes.

If you want general instructions for a specific screen, press `CTRL-W`. If you need help to determine what you should enter in a field on the screen, use the `TAB` key to highlight the field and press `CTRL-F` or `F2`.

Some of the menus display ellipses (...) on the far right or left side. The ellipses indicate that you can move in the direction of the dots, using the arrow keys or `SPACEBAR`, to view other options.

Executing Shell Commands Within ON-Monitor

To execute a shell command from within ON–Monitor, type an exclamation point (`!`) followed by the command. For example, to list the files in the current directory, type the following command:

```
!ls
```

ON-Monitor Screen Options

Table 11-1. Status Menu

Menu	Description
Profile	Displays database server performance statistics
Userthreads	Displays the status of active user threads
Spaces	Displays status information about database server storage spaces and chunks
Databases	Displays the name, owner, and logging mode of the 100 first databases
Logs	Displays status information about the physical-log buffer, the physical log, the logical-log buffer, and the logical-log files
Archive	Displays a list of all backup tapes and logical-log files that you require to restore data using **ontape**
data-Replication	Displays High-Availability Data-Replication (HDR) status and configuration
Output	Stores the output of other status information in a specified file
Configuration	Copies the current database server configuration to a file

Table 11-2. Parameters Menu

Menu	Description
Initialize	Initializes database server disk space or modifies disk-space parameters
Shared-Memory	Initializes database server shared memory or modifies shared-memory parameters
perFormance	Specifies the number of virtual processors for each VP class
data-Replication	Specifies the HDR parameters
diaGnostics	Specifies values for the diagnostics parameters
pdQ	Changes parameters for parallel database queries
Add-Log	Adds a logical-log file to a dbspace
Drop-Log	Drops a logical-log file from a dbspace
Physical-Log	Changes the size or the location of the database server physical log

Table 11-3. Dbspaces Menu

Menu	Description
Create	Creates a dbspace
BLOBSpace	Creates a blobspace
Mirror	Adds mirroring to an existing storage space or ends mirroring for a storage space
Drop	Drops a storage space from the database server configuration
Info	Displays the identification number, location, and fullness of each chunk assigned to a storage space
Add_chunk	Adds a chunk to a storage space
datasKip	Changes the database parameter

Table 11-3. Dbspaces Menu (continued)

Menu	Description
Status	Changes the status of a chunk in a mirrored pair

Table 11-4. Mode Menu

Menu	Description
Startup	Initializes shared memory and takes the database server to quiescent mode
On-Line	Takes the database server from quiescent to online mode
Graceful-Shutdown	Takes the database server from online to quiescent mode so users can complete work
Immediate-Shutdown	Takes the database server from online to quiescent mode in 10 seconds
Take-Offline	Detaches shared memory and immediately takes the database server to offline mode
Add-Proc	Adds virtual processors
Drop-Proc	Drops virtual processors
deCision-support	Sets decision-support parameters dynamically
Single-user	Tells the server to change into single-user mode

Table 11-5. Force-Ckpt Menu

Menu	Description
Force-Ckpt	Displays the time of the most-recent checkpoint or forces the database server to execute a checkpoint

Table 11-6. Archive Menu

Menu	Description
Tape-Parameters	Modifies the **ontape** parameters for the backup tape device

Table 11-7. Logical Logs Menu

Menu	Description
Databases	Modifies the logging status of a database
Tape-Parameters	Modifies the **ontape** parameters for the logical-log backup tape device

Setting Configuration Parameters in ON-Monitor

Figure 11-1 shows which ONCONFIG parameters correspond to the Initialization screen.

```
                                  DISK PARAMETERS
Page Size     [  2] Kbytes                             Mirror {MIRROR}

Tape Dev.     {TAPEDEV}
Block Size    {TAPEBLK}                    Total Tape Size  {TAPESIZE}
Log Tape Dev. {LTAPEDEV}
Block Size    {LTAPEBLK}                   Total Tape Size {LTAPESIZE}
Stage Blob    {STAGEBLOB}

Root Name     {ROOTNAME}                          Root Size {ROOTSIZE}
Primary Path  {ROOTPATH}
                                              Root Offset {ROOTOFFSET}
Mirror Path   {MIRRORPATH}
                                          Mirror Offset {MIRROROFFSET}
Phy. Log Size {PHYSFILE}                       Log. Log Size {LOGSIZE}
                                     Number of Logical Logs {LOGFILES}
```

Figure 11-1. Initialization Screen with Parameter Names

Figure 11-2 shows which ONCONFIG parameters correspond to the Shared-Memory screen.

```
                         SHARED MEMORY PARAMETERS
 Server Number           {SERVERNUM}           Server Name       {DBSERVERNAME}
 Server Aliases          {DBSERVERALIASES}
 Dbspace Temp            {DBSPACETEMP}
 Deadlock Timeout        {DEADLOCK_TIMEOUT}   Dbspace Down Option   {ONDBSPACEDOWN}
 Forced Residency        {RESIDENCY}           Number of Page Cleaners {CLEANERS
 Non Res. SegSize (K)    {SHMVIRTSIZE}            Stack Size (K)     {STACKSIZE}
 Heterogeneous Commit    {HETERO_COMMIT}   Optical Cache Size (K)   {OPCACHEMAX}
 Physical Log Buffer Size {PHYSBUFF}       Transaction Timeout        {TXTIMEOUT}
 Logical Log Buffer Size  {LOGBUFF}        Index Page Fill Factor    {FILLFACTOR}
 Max # of Locks          {LOCKS}            Add SegSize                  {SHMADD}
 Max # of Buffers        {BUFFERS}          Total Memory               {SHMTOTAL}

Resident Shared Memory size  [     ] Kbytes             Page Size  [   2] Kbytes
```

Figure 11-2. Shared-Memory Screen with Parameter Names

Note: Although Dynamic Server can support a shared memory segment that is larger than 4 gigabytes, ON-Monitor does not support a shared memory segment that is larger than 4 gigabytes. Therefore, the ON-Monitor screen cannot hold values that are larger than 4 gigabytes.

Figure 11-3 shows which ONCONFIG parameters correspond to the Performance Tuning screen.

```
                      PERFORMANCE TUNING PARAMETERS

Multiprocessor Machine   {MULTIPROCESSOR}    LRU Max Dirty       {LRU_MAX_DIRTY}
Num Procs to Affinity    {VPCLASS aff}       LRU Min Dirty       {LRU_MIN_DIRTY}
Proc num to start with   {VPCLASS num}       Checkpoint Interval     {CKPTINTVL}
                                             Num of Read Ahead Pages  {RA_PAGES}
CPU VPs                  {VPCLASS cpu}       Read Ahead Threshold {RA_THRESHOLD}
AIO VPs                  {VPCLASS aio}
Single CPU VP            {SINGLE_CPU_VP}     NETTYPE settings:
Use OS Time              {USE_OS_TIME}        Protocol Threads Users VP-class
Disable Priority Aging   {VPCLASS noage}      [      ] [    ] [  ] [     ]
Offline Recovery Threads {OFF_RECVRY_THREADS}
Online Recovery Threads  {ON_RECVRY_THREADS}
Num of LRU queues {LRUS}
```

Figure 11-3. Performance Screen with Parameter Names

Figure 11-4 shows which ONCONFIG parameters correspond to the Data Replication screen.

```
                       DATA REPLICATION PARAMETERS

Interval       {DRINTERVAL}
Timeout        {DRTIMEOUT}
Lost & Found   {DRLOSTFOUND}
```

Figure 11-4. Data-Replication Screen with Parameter Names

Figure 11-5 shows which ONCONFIG parameters correspond to the Diagnostics screen.

```
                       DIAGNOSTIC PARAMETERS

Message Log              {MSGPATH}
Console Msgs.            {CONSOLE}
Alarm Program            {ALARMPROGRAM}

Dump Shared Memory       {DUMPSHMEM}
Dump Gcore               {DUMPGCORE}
Dump Core                {DUMPCORE}
Dump Count               {DUMPCNT}
Dump Directory           {DUMPDIR}
```

Figure 11-5. Diagnostics Screen with Parameter Names

Figure 11-6 shows which ONCONFIG parameters correspond to the PDQ screen.

```
PARALLEL DATABASE QUERIES PARAMETERS

Max PDQ Priority                  {MAX_PDQPRIORITY}
Decision Support Queries          {DS_MAX_QUERIES}
Decision Support Memory (Kbytes)  {DS_TOTAL_MEMORY}
Maximum Decision Support Scans    {DS_MAX_SCANS}
Dataskip                          {DATASKIP}
Optimizer Hint                    {OPTCOMPIND}
Non PDQ Memory                    {DS_NONPDQ_QUERY_MEM}
```

Figure 11-6. PDQ Screen with Parameter Names

Figure 11-7 shows the ON-Monitor screen for creating a dbspace.

```
                                                  CREATE DBSPACE

Dbspace Name [                  ]     Mirror [ ]     Temp [ ]
Page Size [   2] Kbytes
PRIMARY CHUNK INFORMATION:
 Full Pathname [                                                  ]
            Offset [        0] Kbytes        Size [        0] Kbytes
     MIRROR CHUNK INFORMATION:
     Full Pathname [                                                  ]
           Offset [        0] Kbytes
```

Figure 11-7. Create Dbspace Screen

Note: All tables, indexes, and other allocations within the dbspace use pages of the specified page size. The value for `Page Size` must be a multiple of the page size of the root dbspace.

Chapter 12. The onparams Utility

In This Chapter

The **onparams** flags determine which of the following operations **onparams** performs. Any **onparams** command fails if a storage-space backup is in progress. If you do not use any options, **onparams** returns a usage statement.

Function	onparams Command	Database Server Mode
Add a logical-log file	**onparams -a -d dbspace [-i]**	Online, quiescent, or fast-recovery mode
Drop a logical-log file	**onparams -d -l** *lognum*	Online, quiescent, or fast-recovery mode
Change the size or location of the physical log	**onparams -p**	Quiescent mode only
Add a new buffer pool	**onparams -b**	Online, quiescent, or single-user mode

On UNIX, you must be logged in as user **root** or user **informix** to execute **onparams**. On Windows, you must be a member of the **Informix-Admin** group.

Syntax

```
>>--onparams--+-----------------------------------------+---------><
              |                              (1)        |
              +-| Add a Logical-Log File |--------------+
              |                               (2)       |
              +-| Drop a Logical-Log File |-------------+
              |                                      (3)|
              +-| Change Physical-Log  Parameters |-----+
              |                           (4)           |
              '-| Add a New Buffer Pool |---------------'
```

Notes:

1 See page 12-2

2 See page 12-2

3 See page 12-3

4 See page 12-4

Add a Logical-Log File

Add a Logical-Log File:

```
├──-a──-d──dbspace──┬──────────┬──┬────┬──────────────────────────────────┤
                    └─-s──size─┘  └─-i─┘
```

Element	Purpose	Key Considerations
-a -d *dbspace*	Adds a logical-log file to the end of the log-file list to the specified *dbspace*	**Additional Information**: You can add a log file to a dbspace only if the database server has adequate contiguous space. The newly added log files have a status of **A** and are immediately available for use. You can add a log file during a backup. You can have a maximum of 32,767 logical-log files. Use **onstat -l** to view the status of your logical-log files. It is recommended that you take a level-0 backup of the root dbspace and the dbspace that contains the log file as soon as possible. **Restrictions:** You cannot add a log file to a blobspace or sbspace. **References:** Syntax must conform to the Identifier segment; see *IBM Informix Guide to SQL: Syntax.*
-i	Inserts the logical-log file after the current log file	Use this option when the Log File Required alarm prompts you to add a logical-log file.
-s *size*	Specifies a size in kilobytes for the new logical-log file	**Restrictions**: This value must be an unsigned integer greater than or equal to 200 kilobytes. **Additional Information**: If you do not specify a size with the **-s** option, the size of the log file is taken from the value of the LOGSIZE parameter in the ONCONFIG file when database server disk space was initialized. **References**: For information on changing LOGSIZE, see the chapter on managing logical-log files in the *IBM Informix Administrator's Guide*.

Drop a Logical-Log File

Drop a Logical- Log File:

```
├──-d──-l──lognum──┬────┬──────────────────────────────────────────────┤
                   └─-y─┘
```

Element	Purpose	Key Considerations
-d -l *lognum*	Allows you to drop a logical-log file specified by the log file number	**Restrictions**: This value must be an unsigned integer greater than or equal to 0. The database server requires a minimum of three logical-log files at all times. You cannot drop a log file if the database server is configured for three logical-log files. Drop log files one at a time. **Additional Information**: You can obtain the ***lognum*** from the **number** field of **onstat -l**. The sequence of ***lognum*** might be out of order. You can drop a log file immediately that has a status of newly Added (**A**). If you drop a log file that has a status of Used (**U**) or Free (**F**), the database server marks it as Deleted (**D**) and drops it when you take a level-0 backup of all the dbspaces.
-y	Causes the database server to automatically respond yes to all prompts	None.

When you move logical-log files to another dbspace, use the **onparams** commands to add and drop logical-log files. See moving a logical-log file, in the chapter on managing logical-log files in the *IBM Informix Administrator's Guide*.

Change Physical-Log Parameters

Change Physical-Log Parameters:

```
          .---------------------.
          V              (1)    |
>>--p-------+--s--size-------+--+--+------+------------------------><
            |           (1)  |     '--y---'
            '--d--dbspace----'
```

Notes:

1 Only one occurrence of this item allowed

Element	Purpose	Key Considerations
-p	Changes the location or size of the physical-log	**Additional Information:** You can use **onparams -p** with **-s**, **-d**, or both. The database server must be quiescent.
-d *dbspace*	Changes the location of the physical-log to the specified *dbspace*	**Additional Information**: The space allocated for the physical log must be contiguous. **References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax.*
-s *size*	Changes the size (in kilobytes) of the physical log	**Restrictions**: This value must be an unsigned integer greater than or equal to 200 kilobytes. **Warning**: If you move the log to a dbspace without adequate contiguous space or increase the log size beyond the available contiguous space, a fatal shared-memory error occurs when you attempt to restart the database server with the new value.
-y	Causes the database server to automatically respond yes to all prompts	None.

Backing Up After You Change the Physical-Log Size or Location

Changes to the physical log do not take effect until you restart the database server. To restart the database server immediately, execute the **onparams** command with the **-y** option.

Create a level-0 backup of the root dbspace immediately after you restart the database server. This backup is critical for proper recovery of the database server.

Changing the Size of the Physical Log and Using Non-Default Page Sizes

If you use non-default page sizes, you might need to increase the size of your physical log. If you perform many updates to non-default pages you might need a 150 to 200 percent increase of the physical log size. Some experimentation might be needed to tune the physical log. You can adjust the size of the physical log as necessary according to how frequently the filling of the physical log triggers checkpoints.

Using a Text Editor to Change the Physical-Log Size or Location

Another way to change the size or location of the physical-log is to edit the ONCONFIG file and restart the database server. For information on changing the physical-log location and size, see the chapter on managing the physical-log in the *IBM Informix Administrator's Guide*.

Add a New Buffer Pool

Add a New Buffer Pool:

```
|---b---g--size--+-------------+--+-------------+--+-----------------+------->
                 '--n--number--'  '--r--number--'  '--x--percentage--'

>--+-----------------+---------------------------------------------------|
   '--m--percentage--'
```

Element	Purpose	Key Considerations
-b	Creates a new buffer pool	**Additional Information:** You can add a new buffer pool while the database server is running. For more information on buffer pools, see the description of the configuration parameter "BUFFERPOOL" on page 1-14 and the information on buffer pools in the *IBM Informix Administrator's Guide*.

Element	Purpose	Key Considerations
-g *size*	Specifies the size in kilobytes of the buffer pages to create	**Additional Information:** Each dbspace you create with a non-default page size must have a corresponding buffer pool with the corresponding page size. If you create a dbspace with a page size that has no buffer pool, the system will automatically create a buffer pool using the fields in the default line of the BUFFERPOOL parameter. **Restrictions:** The size of the buffer pages must be between 2 and 16 kilobytes and it must be a multiple of the default page size.
-m *percent*	Specifies the percentage of modified pages in the LRU queues at which page cleaning is no longer mandatory	**Additional Information:** Fractional values are allowed. If you do not specify this option, the percentage used is the value of the *lru_min_dirty* field as set in the default line of the BUFFERPOOL configuration parameter. For the range of values , see "The lru_min_dirty Field" on page 1-18.
-n *number*	Specifies the number of buffers in the buffer pool	**Additional Information:** If you do not specify this option, the number used is the value of *buffers* as set in the default line of the BUFFERPOOL configuration parameter. For the range of values , see "The buffers Field" on page 1-16.
-r *number*	Specifies the number of LRU (least-recently-used) queues in the shared-memory buffer pool	**Additional Information:** If you do not include this option, the number of LRU queues allocated is equal to the value of *lrus* as set in the default line of the BUFFERPOOL configuration parameter. For the range of values, see "The lrus Field" on page 1-16.
-x *percent*	Specifies the default percentage of modified pages in the LRU queues at which the queue is cleaned	**Additional Information:** Fractional values are allowed. If you do not specify this option, the percentage used is the value of *lru_max_dirty* as set in the default line of the BUFFERPOOL configuration parameter. For the range of values , see "The lru_max_dirty Field" on page 1-18.

Create a buffer pool that corresponds to the page size of the dbspace. It is recommended that you do this before you create the dbspace. You cannot reduce or increase the number of buffers in an existing buffer pool while the database server is running. You also cannot drop a buffer pool while the database server is running. You can, however, add new buffer pools with a new size while the database server is running.

Buffer pools added with the **onparams** utility are put into virtual memory, not into resident memory. Upon restart, buffer pool entries will go into resident memory depending on the amount of memory that is available.

When you add a new buffer pool with the **onparams** utility or when you add a dbspace with a different page size (with the **onspaces** utility), the settings for the BUFFERPOOL configuration parameter in the ONCONFIG file are rewritten to reflect the new entry.

Examples of onparams Commands

The following are examples of **onparams** commands:

```
onparams -a -d rootdbs -s 1000  # adds a 1000-KB log file to rootdbs

onparams -a -d rootdbs -i       # inserts the log file after the current log

onparams -d -l 7                # drops log 7

onparams -p -d dbspace1 -s 3000 # resizes and moves physical-log to dbspace1

onparams -b -g 6 -n 3000 -r 2 -x 2.0 -m 1.0 # adds 3000 buffers of size
6K bytes each with 2 LRUS with maximum dirty of 2% and minimum dirty of 1%
```

Chapter 13. Managing Storage Spaces with the onspaces Utility

In This Chapter

You can perform the following tasks with the **onspaces** utility:

- Create a dbspace or temporary dbspace.
- Create a blobspace.
- Create an extspace.
- Create an sbspace or temporary sbspace.
- Change sbspace default specifications.
- Clean up stray smart large objects in sbspaces.
- Drop a dbspace, blobspace, sbspace, or extspace.
- Rename a dbspace, blobspace, sbspace, or extspace.
- Add a chunk to a dbspace or blobspace.
- Add a chunk to an sbspace.
- Drop a chunk in a dbspace, blobspace, or sbspace.
- Start mirroring.
- End mirroring.
- Change status of a mirrored chunk.
- Specify the DATASKIP parameter.

When you use **onspaces** or ISA to manage a storage space, the database server updates information about the space in the **oncfg_*servername.servernum*** file. For more information on the **oncfg*** file, refer to Appendix A, "Files That the Database Server Uses," on page A-1.

You can specify a maximum of 2047 chunks for a storage space, and a maximum of 2047 storage spaces on the database server system. The storage spaces can be any combination of dbspaces, blobspaces, and sbspaces.

On UNIX, you must be logged in as user **root** or user **informix** to execute **onspaces**. On Windows, you must be a member of the **Informix-Admin** group.

onspaces Syntax

```
>>--onspaces--+---------------------------------------------------+----><
              |                                      (1)          |
              +-| Create a Dbspace or Temporary Dbspace |---------+
              |                   (2)                             |
              +-| Create a Blobspace |-------------------------------+
              |                   (3)                             |
              +-| Create an Extspace |-------------------------------+
              |                                        (4)        |
              +-| Create an Sbspace or Temporary Sbspace |--------+
              |                                        (5)        |
              +-| Change Sbspace Default Specifications, |--------+
              |                                    (6)            |
              +-| Clean Up Stray Smart Large Objects |------------+
              |                                                (7)|
              +-| Drop a Dbspace, Blobspace, Sbspace, or Extspace |-+
              |                                                  (8)
              +-| Rename a Dbspace, Blobspace, Sbspace, or Extspace |+
              |                                      (9)          |
              +-| Add a Chunk to a Dbspace or Blobspace |---------+
              |                          (10)                     |
              +-| Add a Chunk to an Sbspace |---------------------+
              |                                                (11)
              +-| Drop a Chunk in a Dbspace, Blobspace or Sbspace |-+
              |                  (12)                             |
              +-| Start Mirroring |-------------------------------+
              |                (13)                               |
              +-| End Mirroring |---------------------------------+
              |                                     (14)          |
              +-| Change Status of a Mirrored Chunk |-------------+
              |                     (15)                          |
              '-| Specify DATASKIP |------------------------------'
```

Notes:

1 See page 13-3

2 See page 13-7

3 See page 13-8

4 See page 13-9

5 See page 13-14

6 See page 13-15

7 See page 13-16

8 See page 13-17

9 See page 13-18

10 See page 13-19

11 See page 13-21

12 See page 13-22

13 See page 13-24

14 See page 13-24

15 See page 13-25

Create a Dbspace or Temporary Dbspace

Create a Dbspace or Temporary Dbspace:

```
├──-c──-d──dbspace──┬─-p──pathname──────────────────┬─-o──offset──-s──size─────────────────────────────►
                    │                           (1) │
                    └─-p──\──\──.──\──drive─────────┘

►─┬─────────────────────────────────────────┬──┬─────────────────────────────────┬──┬─────────────────┬───────┤
  ├─-ef──extentsize──-en──extentsize────────┤  ├─-m──pathname offset─────────────┤  └─-k──pagesize───┘
  └─-t──────────────────────────────────────┘  │                             (1) │
                                               └─-m──\──\──.──\──drive offset────┘
```

Notes:

1 Windows Only

Element	Purpose	Key Considerations
-c	Creates a dbspace You can create up to 2047 storage spaces of any type.	**Additional Information:** After you create a storage space, you must back up both this storage space and the root dbspace. If you create a storage space with the same name as a deleted storage space, perform another level-0 backup to ensure that future restores do not confuse the new storage space with the old one. **References**: For more information, see creating a dbspace in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

drive	Specifies the Windows drive to allocate as unbuffered disk space
	
	The format can be either **\\.\<*drive*>**, where ***drive*** is the drive letter assigned to a disk partition, or **\\.*PhysicalDrive*<*number*>**, where ***PhysicalDrive*** is a constant value and ***number*** is the physical drive number.
		References: For information on allocating unbuffered disk space, see allocating unbuffered disk space on Windows in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.Examples:
		`\\.\F:`
		`\\.\PhysicalDrive2`
		
		References: For pathname syntax, see your operating-system documentation.

Element	Purpose	Key Considerations
-d ***dbspace***	Names the dbspace to be created	**Restrictions**: The dbspace name must be unique and cannot exceed 128 characters. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the **$** character. **References**: For more information, see creating a dbspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*. The syntax must conform to the Identifier segment. For more information, see the *IBM Informix Guide to SQL: Syntax*.
-ef *extentsize*	Indicates, in kilobytes, the size of the next extents in the tblspace **tblspace**	**Restrictions:** The minimum size of the next extents for the tblspace tblspace of a non-root dbspace is equivalent to 4 dbspace pages, specified in K. For example: 8 KB for a 2 KB page size dbspace, 16 KB for a 4 KB page size dbspace, 32 KB for an 8 KB page size dbspace. The default size for a next extent is 50 dbspace pages. The maximum size of a tblspace tblspace extent is 1048572 pages. On a 2 KB pagesize system this would evaluate to approximately 2 GB. If there is not enough space for a next extent in the primary chunk, the extent is allocated from another chunk. If the specified space is not available, the closest available space is allocated. References: For more information, see specifying first and next extent size in the chapter on managing dbspaces in the *IBM Informix Administrator's Guide*.
-en *extentsize*	Indicates, in kilobytes, the size of the first extent for the tblspace **tblspace**	**Restrictions**: The minimum, and default, size of the first extent for the tblspace tblspace of a non-root dbspace is equivalent to 50 dbspace pages, specified in K. For example: 100 KB for a 2 KB page size dbspace, 200 KB for a 4 KB page size dbspace, 400 KB for an 8 KB page size dbspace. The maximum size of a tblspace tblspace extent is 1048575 pages minus the space needed for any system objects. On a 2 KB pagesize system this would evaluate to approximately 2 GB. References: For more information, see specifying first and next extent size in the chapter on managing dbspaces in the *IBM Informix Administrator's Guide*.

Element	Purpose	Key Considerations
-k *pagesize*	Indicates in kilobytes, the non-default page size for the new dbspace. For systems with sufficient storage, performance advantages of a larger page size can include the following: • Reduced depth of B-tree indexes, even for smaller index keys • You can group on the same page long rows that currently span multiple pages of the default page size • Checkpoint time is typically reduced with larger pages • You can define a different page size for temporary tables, so that they have a separate buffer pool.	**Restrictions:** The page size must be between 2KB and 16KB and must be a multiple of the default page size. For example, if the default page size is 2KB, then *pagesize* can be 2, 4, 6, 8, 10, 12, 14, or 16. If the default page size is 4KB (Windows), then *pagesize* can be 4, 8, 12, or 16. **References**: For more information, see creating a dbspace with a non-default page size in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-m ***pathname offset***	Specifies an optional pathname and offset to the chunk that mirrors the initial chunk of the new dbspace Also see the entries for **-p** ***pathname*** and **-o** ***offset*** in this table.	**References**: For more information, see creating a dbspace in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-o ***offset***	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new dbspace	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The offset must be a multiple of the page size. The maximum offset is 2 or 4 gigabytes, depending on the platform. **References**: For more information, see allocating raw disk space, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

-p ***pathname***	Indicates the disk partition or device of the initial chunk of the dbspace that you are creating	**Additional Information**: The chunk must be an existing unbuffered device or buffered file. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server.
		UNIX example (unbuffered device):
		`/dev/rdsk/c0t3d0s4`
		UNIX example (buffered device):
		`/ix/ids9.2/db1chunkWindows`
		example:`c:\Ifmxdata\ol_icecream\mychunk1.dat`
		
		References: For pathname syntax, see your operating-system documentation.

Element	Purpose	Key Considerations
-s *size*	Indicates, in kilobytes, the size of the initial chunk of the new dbspace	**Restrictions**: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 terabytes, depending on the platform.
-t	Creates a temporary dbspace for storage of temporary tables	**Restrictions:** You cannot mirror a temporary dbspace. You cannot specify the first and next extent sizes for the tblspace **tblspace** of a temporary dbspace. You cannot specify a non-default page size for a temporary dbspace. **References**: For more information, see temporary dbspaces, in the chapter on data storage, and creating a temporary dbspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

Creating a Temporary Dbspace with the -t Option

When you create a temporary dbspace with **onspaces**, the database server uses the newly created temporary dbspace, after you perform the following steps:

- Add the name of the new temporary dbspace to your list of temporary dbspaces in the DBSPACETEMP configuration parameter, the DBSPACETEMP environment variable, or both.
- Restart the database server.

Specifying First and Next Extent Size for the tblspace tblspace

You cannot specify the first and next extent of a temporary dbspace. The extent size for temporary dbspaces is 100 kilobytes for a 2 kilobyte page system or 200 kilobytes for a 4 kilobyte page system.

To specify the first and next extent sizes of a root tblspace **tblspace**, use the TBLTBLFIRST and TBLTBLNEXT configuration parameters before you create the root dbspace the first time you start the database server.

Specifying a Non-Default Page Size with the Same Size as the Buffer Pool

When you create a dbspace with a non-default page size, you must also create a buffer pool specific to that page size. It is recommended that you create the buffer pool before you create the dbspace. Use the **onparams** utility to create a buffer pool. For more information, see "Add a New Buffer Pool" on page 12-4.

When you add a dbspace with a different page size with the **onspaces** utility or you add a new buffer pool (with the **onparams** utility), a new BUFFERPOOL line is appended in the BUFFERPOOL configuration parameter in the ONCONFIG file to reflect the new entry and it is rewritten to disk.

Notes:

1. You cannot change the page size of a dbspace after you create it.
2. You cannot store logical or physical logs in a dbspace that is not the default platform page size.

3. If a dbspace is created when a buffer pool with that page size does not exist, Dynamic Server creates a buffer pool using the values of the fields of the `default` line of the BUFFERPOOL parameter. You cannot have multiple buffer pools with the same page size.

Create a Blobspace

Create a Blobspace:

```
├──-c──-b──blobspace──-g──pageunit──┬─-p──pathname──────────────────┬─-o──offset──-s──size──────────►
                                     │                       (1)     │
                                     └─-p──\──\──.──\──drive─────────┘

►──┬──────────────────────────────┬─────────────────────────────────────────────────────────────┤
   ├─-m──pathname offset──────────┤
   │                       (1)    │
   └─-m──\──\──.──\──drive offset─┘
```

Notes:

1 Windows Only

Element	Purpose	Key Considerations
-b ***blobspace***	Names the blobspace to be created	**Restrictions**: The blobspace name must be unique and cannot exceed 128 characters. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the **$** character. **References**: For more information, see creating a blobspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*. The syntax must conform to the Identifier segment. For more information, see the *IBM Informix Guide to SQL: Syntax*.
-**c**	Creates a dbspace, blobspace, sbspace, or extspace You can create up to 2047 storage spaces of any type.	**Additional Information:** After you create a storage space, you must back up both this storage space and the root dbspace. If you create a storage space with the same name as a deleted storage space, perform another level-0 backup to ensure that future restores do not confuse the new storage space with the old one. **References**: For more information, see creating a dbspace, blobspace, or extspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

drive	Specifies the Windows drive to allocate as unbuffered disk space
	
	The format can be either **\\\\.\\<*drive*>**, where ***drive*** is the drive letter assigned to a disk partition, or **\\\\.*PhysicalDrive*<*number*>**, where ***PhysicalDrive*** is a constant value and ***number*** is the physical drive number.
		References: For information on allocating unbuffered disk space, see allocating unbuffered disk space on Windows in the chapter on managing disk space in the *IBM Informix Administrator's Guide*. Examples:
		`\\.\F:`
		`\\.\PhysicalDrive2`
		
		References: For pathname syntax, see your operating-system documentation.

Element	Purpose	Key Considerations
-g ***pageunit***	Specifies the blobspace blobpage size in terms of ***page_unit***, the number of disk pages per blobpage	**Restrictions**: Unsigned integer. Value must be greater than 0. **References**: For more information, see blobpage size considerations, in the chapter on I/O Activity in the *IBM Informix Performance Guide*.
-m ***pathname offset***	Specifies an optional pathname and offset to the chunk that mirrors the initial chunk of the new blobspace or dbspace Also see the entries for **-p** ***pathname*** and **-o** ***offset*** in this table.	**References**: For more information, see creating a dbspace or a blobspace in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-o ***offset***	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new blobspace, dbspace, or sbspace	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 2 or 4 gigabytes, depending on the platform. **References**: For more information, see allocating raw disk space, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-p ***pathname***	Indicates the disk partition or device of the initial chunk of the blobspace or dbspace that you are creating	**Additional Information**: The chunk must be an existing unbuffered device or buffered file. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server. UNIX example (unbuffered device): `/dev/rdsk/c0t3d0s4` UNIX example (buffered device): `/ix/ids9.2/db1chunk` Windows example:`c:\Ifmxdata\ol_icecream\mychunk1.dat` **References**: For pathname syntax, see your operating-system documentation.
-s ***size***	Indicates, in kilobytes, the size of the initial chunk of the new blobspace or dbspace	**Restrictions**: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 terabytes, depending on the platform.

Create an Extspace

Create an Extspace:

```
├──-c──-x──extspace──-l──location──-o──offset──-s──size──────────────────────────┤
```

Element	Purpose	Key Considerations
-**c**	Creates a dbspace, blobspace, sbspace, or extspace You can create up to 2047 storage spaces of any type.	**Additional Information:** After you create a storage space, you must back up both this storage space and the root dbspace. If you create a storage space with the same name as a deleted storage space, perform another level-0 backup to ensure that future restores do not confuse the new storage space with the old one. **References**: For more information, see creating a dbspace, blobspace, or extspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-l ***location***	Specifies the location of the extspace The access method determines the format of this string.	**Restrictions**: String. Value must not be longer than 255 bytes. **References**: For more information, see creating an extspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-o ***offset***	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new blobspace, dbspace, or sbspace	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 2 or 4 gigabytes, depending on the platform. **References**: For more information, see allocating raw disk space, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-s ***size***	Indicates, in kilobytes, the size of the initial chunk of the new blobspace or dbspace	**Restrictions**: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 terabytes, depending on the platform.
-x ***extspace***	Names the extspace to be created	**Restrictions**: Extspace names can be up to 128 characters. They must be unique, begin with a letter or underscore, and contain only letters, digits, underscores, or **$** characters. **References**: For more information, see extspaces, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

Create an Sbspace or Temporary Sbspace

Create an Sbspace:

```
├──-c──-S──sbspace──┬────┬──-p──pathname──-o──offset──-s──size──────────►
                    └─-t─┘

►──┬──────────────────────┬──┬──────────────┬──┬────────────────┬────────►
   └─-m──pathname offset──┘  └─-Ms──mdsize──┘  └─-Mo──mdoffset──┘

►──┬──────────────────────┬──────────────────────────────────────────────┤
   └─-Df──default list────┘
```

Element	Purpose	Key Considerations
-S ***sbspace***	Names the sbspace to be created	**Restrictions**: The sbspace name must be unique and must not exceed 128 characters. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the **$** character. **References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*.
-**c**	Creates an sbspace You can create up to 2047 storage spaces of any type.	None.
-m ***pathname offset***	Specifies an optional pathname and offset to the chunk that mirrors the initial chunk of the new sbspace Also see the entries for **-p** ***pathname*** and **-o** ***offset*** in this table.	**References**: For more information, see sbspaces in the chapter on data storage, and creating an sbspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-Mo ***mdoffset***	Indicates, in kilobytes, the offset into the disk partition or into the device where metadata will be stored.	**Restrictions**: Value can be an integer between 0 and the chunk size. You cannot specify an offset that causes the end of the metadata space to be past the end of the chunk. **References**: For more information, see sizing sbspace metadata, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-Ms ***mdsize***	Specifies the size, in kilobytes, of the metadata area allocated in the initial chunk The remainder is user-data space.	**Restrictions**: Value can be an integer between 0 and the chunk size.
-o ***offset***	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the sbspace	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 4 terabytes for systems with a two-kilobyte page size and 8 terabytes for systems with a four-kilobyte page size. **References**: For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-p ***pathname***	Indicates the disk partition or unbuffered device of the initial chunk of the sbspace	**Additional Information**: The chunk must be an existing unbuffered device or buffered file. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server. **References**: For pathname syntax, see your operating-system documentation.
-s ***size***	Indicates, in kilobytes, the size of the initial chunk of the new sbspace	**Restrictions**: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 gigabytes, depending on the platform.

Element	Purpose	Key Considerations
-t	Creates a temporary sbspace for storage of temporary smart large objects. You can specify the size and offset of the metadata area	**Restrictions:** You cannot mirror a temporary sbspace. You can specify any **-Df** option, except the LOGGING=ON option, which has no effect. **References**: For more information, see "Creating a Temporary Sbspace with the -t Option" on page 13-11.
-Df ***default list***	Lists default specifications for smart large objects stored in the sbspace	**Restrictions**: Tags are separated by commas. If a tag is not present, system defaults take precedence. The list must be enclosed in double quotation marks (") on the command line. **References**: For a list of tags and their parameters, see Table 13-1 on page 13-11.

Creating a Temporary Sbspace with the -t Option

This example creates a temporary sbspace of 1000 kilobytes:

```
onspaces -c -S tempsbsp -t -p ./tempsbsp -o 0 -s 1000
```

You can optionally specify the name of the temporary sbspace in the SBSPACETEMP configuration parameter. Restart the database server so that it can use the temporary sbspace.

Creating an Sbspace with the -Df option

When you create an sbspace with the optional **-Df** option, you can specify several default specifications that affect the behavior of the smart large objects stored in the sbspace. The default specifications must be expressed as a list separated by commas. The list need not contain all of the tags. The list of tags must be enclosed in double quotation marks ("). The table in Table 13-1 on page 13-11 describes the tags and their default values.

The four levels of inheritance for sbspace characteristics are system, sbspace, column, and smart large objects. For more information, see smart large objects in the chapter on where data is stored in the *IBM Informix Administrator's Guide*.

Table 13-1. -Df Default Specifications

Tag	Values	Default	Description
ACCESSTIME	ON or OFF	OFF	When set to ON, the database server tracks the time of access to all smart large objects stored in the sbspace. **References**: For information about altering storage characteristics of smart large objects, see the *IBM Informix DataBlade API Programmer's Guide*.

Table 13-1. -Df Default Specifications (continued)

Tag	Values	Default	Description
AVG_LO_SIZE	Windows: 4 to 2**31 UNIX: 2 to 2**31	8	Specifies the average size, in kilobytes, of the smart large object stored in the sbspace The database server uses this value to calculate the size of the metadata area. Do not specify AVG_LO_SIZE and **-Ms** together. You can specify AVG_LO_SIZE and the metadata offset (**-Mo**) together. If the size of the smart large object exceeds 2**31, specify 2**31. If the size of the smart large object is less than 2 on UNIX or less than 4 in Windows, specify 2 or 4. Error 131 is returned if you run out of space in the metadata and reserved areas in the sbspace. To allocate additional chunks to the sbspace that consist of metadata area only, use the **-Ms** option instead. **References**: For more information, see creating smart large objects, in the chapter on managing data on disk in the *IBM Informix Administrator's Guide*.
BUFFERING	ON or OFF	ON	Specifies the buffering mode of smart large objects stored in the sbspace If set to ON, the database server uses the buffer pool in the resident portion of shared memory for smart-large-object I/O operations. If set to OFF, the database server uses light I/O buffers in the virtual portion of shared memory (lightweight I/O operations). **Restriction:** BUFFERING = OFF is incompatible with LOCK_MODE = RANGE and creates a conflict **References**: For more information, see lightweight I/O, in the chapter on configuration effects on memory in the *IBM Informix Performance Guide*.
LOCK_MODE	RANGE or BLOB	BLOB	Specifies the locking mode of smart large objects stored in the sbspace If set to RANGE, only a range of bytes in the smart large object is locked. If set to BLOB, the entire smart large object is locked. **Restriction:** LOCK_MODE = RANGE is incompatible with BUFFERING = OFF and creates a conflict. **References**: For more information, see smart large objects, in the chapter on locking in the *IBM Informix Performance Guide*.

Table 13-1. -Df Default Specifications (continued)

Tag	Values	Default	Description
LOGGING	ON or OFF	OFF	Specifies the logging status of smart large objects stored in the sbspace If set to ON, the database server logs changes to the user data area of the sbspace. When you turn on logging for an sbspace, take a level-0 backup of the sbspace. When you turn off logging, the following message displays: You are turning off smart large object logging. **References**: For more information, see smart large objects, in the chapters on data storage and logging in the *IBM Informix Administrator's Guide*. For information about **onspaces -ch** messages, see Appendix E, "Error Messages," on page E-1.
EXTENT_SIZE	4 to 2**31	None	Specifies the size, in kilobytes, of the first allocation of disk space for smart large objects stored in the sbspace when you create the table Let the system select the EXTENT_SIZE value. To reduce the number of extents in a smart large object, use **mi_lo_specset_estbytes** (DataBlade API) or **ifx_lo_specset_estbytes** (ESQL/C) to hint to the system the total size of the smart large object. The system attempts to allocate a single extent for the smart large object. **References**: For more information, see smart large objects, in the chapter on where data is stored in the *IBM Informix Administrator's Guide*. For information about altering storage characteristics of smart large objects, see the *IBM Informix DataBlade API Programmer's Guide* or the *IBM Informix ESQL/C Programmer's Manual*.
MIN_EXT_SIZE	2 to 2**31	Windows: 4UNIX: 2	Specifies the minimum amount of space, in kilobytes, to allocate for each smart large object The following message displays: Changing the sbspace minimum extent size: old value *value1* new value *value2*. **References**: For information about tuning this value, see smart large objects, in the chapter on configuration effects on I/O utilization in the *IBM Informix Performance Guide*. For information about **onspaces -ch** messages, see Appendix E, "Error Messages," on page E-1.
NEXT_SIZE	4 to 2**31	None	Specifies the extent size, in kilobytes, of the next allocation of disk space for smart large objects when the initial extent in the sbspace becomes full. Let the system select the NEXT_SIZE value. To reduce the number of extents in a smart large object, use **mi_lo_specset_estbytes** or **ifx_lo_specset_estbytes** to hint to the system the total size of the smart large object. The system attempts to allocate a single extent for the smart large object. **References**: For more information, see smart large objects, in the chapter on where data is stored in the *IBM Informix Administrator's Guide*. For information about obtaining the size of smart large objects, see the *IBM Informix DataBlade API Programmer's Guide* or the *IBM Informix ESQL/C Programmer's Manual*.

This example creates a 20-megabyte mirrored sbspace, **eg_sbsp,** with the following specifications:

- An offset of 500 kilobytes for the primary and mirror chunks
- An offset of 200 kilobytes for the metadata area
- An average expected smart-large-object size of 32 kilobytes
- Log changes to the smart large objects in the user-data area of the sbspace

UNIX Only

```
% onspaces -c -S eg_sbsp -p /dev/raw_dev1 -o 500 -s 20000
   -m /dev/raw_dev2 500 -Mo 200 -Df "AVG_LO_SIZE=32,LOGGING=ON"
```

End of UNIX Only

Changing the -Df Settings

As the database server administrator, you can override or change the **-Df** default settings in one of the following ways:

- To change the default settings for an sbspace, use the **onspaces -ch** option. For more information, refer to “Change Sbspace Default Specifications” on page 13-14.
- To override the following **-Df** default settings for a specific table, use the SQL statements CREATE TABLE or ALTER TABLE:
  - LOGGING
  - ACCESSTIME
  - EXTENT_SIZE
  - NEXT_SIZE

  For more information on the ALTER TABLE and CREATE TABLE statements, see the *IBM Informix Guide to SQL: Syntax*.

The programmer can override these **-Df** default settings with DataBlade API and ESQL/C functions. For information about altering storage characteristics of smart large objects, see the *IBM Informix DataBlade API Programmer’s Guide* and the *IBM Informix ESQL/C Programmer’s Manual*.

Using the onspaces -g Option

The **onspaces -g** option is not used for sbspaces. The database server uses a different method to determine the number of pages to transfer in an I/O operation for sbspaces than for blobspaces. The database server can automatically determine the block size to transfer in an I/O operation for smart large objects. For more information, see sbspace extent sizes in the chapter on I/O activity in your *IBM Informix Performance Guide*.

Change Sbspace Default Specifications

Change Sbspace Default Specifications:

```
├──-ch──sbspace──-Df──default list─────────────────────────────────┤
```

Element	Purpose	Key Considerations
-ch	Indicates that one or more sbspace default specifications are to be changed	None.
sbspace	Names the sbspace for which to change the default specifications	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For background information, see changing default specifications of an sbspace with **onspaces** in the *IBM Informix Performance Guide*.
-Df *default list*	Lists new default specifications for smart large objects stored in the sbspace	**Restrictions**: Tags are separated by commas. If a tag is not present, system defaults take precedence. The list must be enclosed in double quotation marks (") on the command line. **References**: For a list of tags and their parameters, see Table 13-1 on page 13-11.

You can change any of the **-Df** tags with the **onspaces -ch** option. The database server applies the change to each smart large object that was created prior to changing the default specification.

For example, to turn off logging for the sbspace that you created in "Creating an Sbspace with the -Df option" on page 13-11, use the following command:

```
onspaces -ch eg_sbsp -Df "LOGGING=OFF"
```

Note: After you turn on logging for an sbspace, take a level-0 backup of the sbspace to create a point from which to recover.

Clean Up Stray Smart Large Objects in Sbspaces

Clean Up Stray Smart Large Objects in Sbspaces:

```
├──-cl──sbspace──────────────────────────────────────────────┤
```

Element	Purpose	Key Considerations
-cl	Cleans up stray smart large objects in an sbspace	To find any stray smart large objects, use the **oncheck -pS** command when no users are connected to the database server. The smart large objects with a reference count of 0 are stray objects.
sbspace	Names the sbspace to be cleaned up	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*.

During normal operation, no unreferenced (stray) smart large objects should exist. When you delete a smart large object, the space is released. If the database server fails or runs out of system memory while you are deleting a smart large object, the smart large object might remain as a stray object.

The following is an example of the **onspaces -cl** command:

```
onspaces -cl myspace
```

The best way to find the reference count for a smart large object is to call the **mi_lo_stat** or **ifx_lo_stat** functions from a C program. Although the

mi_lo_increfcount and **mi_lo_decrefcount** functions return the reference count, they increment or decrement the reference count. For more information on these functions, see the *IBM Informix DataBlade API Function Reference*.

Drop a Dbspace, Blobspace, Sbspace, or Extspace

Drop a Dbspace, Blobspace, Sbspace, or Extspace:

```
├──-d──┬─dbspace──────────┬──┬──────┬──────────────────────────────────┤
       ├─blobspace────────┤  └─-y───┘
       ├─┬──────┬─sbspace─┤
       │ └─-f───┘         │
       └─extspace─────────┘
```

Element	Purpose	Key Considerations
-d	Indicates that a dbspace, blobspace, sbspace, or extspace is to be dropped	**Additional Information:** You can drop a dbspace, blobspace, sbspace, or extspace while the database server is online or in quiescent mode. After you drop a storage space, you must back it up to ensure that the **sysutils** database and the reserved pages are up-to-date. **Restriction**: Execute **oncheck -pe** to verify that no table is currently storing data in the dbspace, blobspace, or sbspace. **References**: For more information, see dropping a storage space, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-y	Causes the database server to automatically respond yes to all prompts	None.
-f	Drops an sbspace that contains user data and metadata	**Additional Information**: You must use the **-f** (force) option to drop an sbspace that contains data. **Restriction:** Use the **-f** option with sbspaces only. **Warning:** If you use the **-f** option, the tables in the database server might have dead pointers to the smart large objects that were deleted with this option. **References:** For more information, see dropping a chunk from an sbspace with **onspaces**, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
blobspace	Names the blobspace to be dropped	**Additional Information**: Before you drop a blobspace, drop all tables that include a TEXT or BYTE column that references the blobspace.
dbspace	Names the dbspace to be dropped	**Additional Information**: Before you drop a dbspace, drop all databases and tables that you previously created in the dbspace.
extspace	Names the extspace to be dropped	**Additional Information**: You cannot drop an extspace if it is associated with an existing table or index.
sbspace	Names the sbspace to be dropped	**Additional Information**: Before you drop an sbspace, drop all tables that include a BLOB or CLOB column that references the sbspace.

Important: Do not specify a pathname when you drop these storage spaces.

Rename a Dbspace, Blobspace, Sbspace, or Extspace

Rename a Space:

```
├──-ren──┬─dbspace───┬──-n──name────────────────────────────────────────┤
         ├─blobspace─┤
         ├─sbspace───┤
         └─extspace──┘
```

Element	Purpose	Key Considerations
-ren	Causes the database server to rename the specified blobspace, dbspace, extspace, or sbspace	**Restrictions**: You can rename a blobspace, dbspace, extspace, or sbspace when the database server is in quiescent mode. For more information, see the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-n *name*	Specifies the new name for the blobspace, dbspace, extspace, or sbspace	**Restrictions**: The blobspace, dbspace, external space, or sbspace name must be unique and cannot exceed 128 characters. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the **$** character. **References**: For more information, see the chapter on managing disk space in the *IBM Informix Administrator's Guide*. The syntax must conform to the Identifier segment. For more information, see the *IBM Informix Guide to SQL: Syntax*.
blobspace	Names the blobspace to be renamed	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see renaming spaces, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
dbspace	Names the dbspace to be renamed	**Restrictions:** You cannot rename a critical dbspace, such as the root dbspace or a dbspace that contains physical logs. **Additional Information:** If you rename dbspaces that are included in the DATASKIP list, update the DATASKIP configuration parameter with the new names using the **onspaces -f** command. **References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see renaming spaces, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
extspace	Names the extspace to be renamed	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see renaming spaces, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
sbspace	Names the sbspace to be renamed	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see renaming spaces, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

Renaming a Dbspace, Blobspace, Sbspace, or Extspace when Enterprise Replication Is Active

You can rename a space (dbspace, blobspace, sbspace, or extspace) when Enterprise Replication is active. When you put the database server into quiescent mode to rename the space, Enterprise Replication will be disconnected. You can then rename the space. The servers will resynchronize after you put the database server into online mode. If you want to rename the same space on another server, you must put that server into quiescent mode and rename the space separately. No enforced relationship is propagated between renamed spaces on different ER servers; the same tables can be in different spaces.

If the Enterprise Replication server also participates in High-Availability Data Replication (HDR), you can rename the dbspace on the primary server and it will be automatically propagate to the secondary server. (The secondary server cannot participate in Enterprise Replication.)

Performing an Archive after Renaming a Space

After renaming any space (except extspaces or temporary spaces), perform a level-0 archive of the renamed space and the root dbspace. This will ensure that you can restore the spaces to a state including or following the rename dbspace operation. It is also necessary prior to performing any other type of archive.

Add a Chunk to a Dbspace or Blobspace

Add a Chunk to a Dbspace or Blobspace:

```
├──-a──┬─dbspace───┬──┬─-p──pathname─────────────────┬──-o──offset──-s──size──────►
       └─blobspace─┘  │                         (1)  │
                      └─-p──\──\──.──\──drive────────┘

►──┬─-m──pathname offset───────────────┬─────────────────────────────────────────┤
   │                               (1) │
   └─-m──\──\──.──\──drive offset──────┘
```

Notes:

1 Windows Only

Element	Purpose	Key Considerations
-a	Indicates that a chunk is to be added	Additional Information: A dbspace, blobspace, or sbspace can contain up to 2047 chunks.
drive	Specifies the Windows drive to allocate as unbuffered disk space The format can be either **\\.\<*drive*>**, where ***drive*** is the drive letter assigned to a disk partition, or **\\.*PhysicalDrive*<*number*>**, where ***PhysicalDrive*** is a constant value and ***number*** is the physical drive number.	**References:** For more information on allocating unbuffered disk space, see allocating raw disk space on Windows in the chapter on managing disk space in the *IBM Informix Administrator's Guide*. Example: `\\.\F:` **References**: For pathname syntax, see your operating-system documentation.
-m ***pathname offset***	Specifies an optional pathname and offset to the chunk that mirrors the new chunk Also see the entries for *pathname* and *offset* in this table.	**References**: For more information, see adding a chunk to a dbspace and adding a chunk to a blobspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

Element	Purpose	Key Considerations
-o ***offset***	After the **-a** option, *offset* indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new blobspace or dbspace	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes. **References**: For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

-p ***pathname***	Indicates the disk partition or unbuffered device of the initial chunk of the blobspace or dbspace that you are adding
	
	The chunk must be an existing unbuffered device or buffered file.
		Additional Information: The chunk name can be up to 128 characters. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server.
		UNIX example (unbuffered device):
		`/dev/rdsk/c0t3d0s4`
		UNIX example (buffered device):
		`/ix/ids9.2/db1chunk`
		Windows example:
		`c:\Ifmxdata\ol_icecream\mychunk1.dat`
		
		References: For pathname syntax, see your operating-system documentation.

-s ***size***	Indicates, in kilobytes, the size of the new blobspace or dbspace chunk	**Restrictions**: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes.
blobspace	Names the blobspace to which you are adding a chunk	**Restrictions**: See adding a chunk to a blobspace in the chapter on managing disk space in the *IBM Informix Administrator's Guide*. **References:** Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*.
dbspace	Names the dbspace to which you are adding a chunk	**Restrictions**: See adding a chunk to a dbspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*. **References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*.

Add a Chunk to an Sbspace

Add a Chunk to an Sbspace:

```
|---a--sbspace---p--pathname---o--offset---s--size--+------------------------+---->
                                                     '--m--pathname offset--'

>--+------------+--+--------------+--+----+-------------------------------------|
   '--Ms--mdsize-'  '--Mo--mdoffset-'  '--U-'
```

Element	Purpose	Key Considerations
-a	Indicates that a chunk is to be added	Additional Information: An sbspace can contain up to 32,766 chunks.

Element	Purpose	Key Considerations
-m ***pathname offset***	Specifies an optional pathname and offset to the chunk that mirrors the new chunk Also see the entries for ***pathname*** and ***offset*** in this table.	**References**: For background information, see adding a chunk to an sbspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-Mo ***mdoffset***	Indicates, in kilobytes, the offset into the disk partition or into the device where metadata should be stored	**Restrictions**: Value can be an integer between 0 and the chunk size. You cannot specify an offset that causes the end of the metadata space to be past the end of the chunk. **References**: For background information, see sizing sbspace metadata, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-Ms ***mdsize***	Specifies the size, in kilobytes, of the metadata area allocated in the initial chunk. The remainder is user-data space	**Restrictions**: Value can be an integer between 0 and the chunk size. **References**: For background information, see sizing sbspace metadata, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-o ***offset***	After the **-a** option, *offset* indicates, in kilobytes, the offset into the disk partition or into the unbuffered device to reach the initial chunk of the new blobspace or dbspace.	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 2 or 4 gigabytes, depending on the platform. **References**: For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-p ***pathname***	Indicates the disk partition or unbuffered device of the initial chunk of the sbspace that you are creating The chunk must be an existing unbuffered device or buffered file.	**Additional Information**: The chunk name can be up to 128 characters. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server. **References**: For pathname syntax, see your operating-system documentation.
-U	Specifies that the entire chunk should be used to store user data	**Restrictions**: The **-M** and **-U** options are mutually exclusive. **References**: For background information, see adding a chunk to an sbspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-s ***size***	Indicates, in kilobytes, the size of the new sbspace chunk	**Restrictions**: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes.
sbspace	Names the sbspace to which you are adding a chunk	**Restrictions**: See adding a chunk to an sbspace in the chapter on managing disk space in the *IBM Informix Administrator's Guide*. **References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*.

Drop a Chunk in a Dbspace, Blobspace, or Sbspace

Drop a Chunk:

```
|---d--+-dbspace-------+--p--pathname---o--offset--+------+--------------|
       +-blobspace-----+                           '--y---'
       '-+------+-sbspace-'
         '--f---'
```

Element	Purpose	Key Considerations
-d	Drops a chunk	**Restrictions**: You can drop a chunk from a dbspace, temporary dbspace, or sbspace when the database server is online or quiescent. For more information, see the chapter on managing disk space in the *IBM Informix Administrator's Guide*. You can drop a chunk from a blobspace only when the database server is in quiescent mode.
-f	Drops an sbspace chunk that contains user data but *no* metadata If the chunk contains metadata for the sbspace, you must drop the entire sbspace.	**Restrictions:** Use the **-f** option with sbspaces only. If you omit the **-f** option, you cannot drop an sbspace that contains data. **References:** For more information, see dropping a chunk from an sbspace with **onspaces**, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-o ***offset***	Indicates, in kilobytes, the offset into the disk partition or into the unbuffered device to reach the initial chunk of the dbspace, blobspace, or sbspace that you are dropping	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes. **References**: For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-p ***pathname***	Indicates the disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that you are dropping	**Additional Information**: The chunk must be an existing unbuffered device or buffered file. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server. **References**: For pathname syntax, see your operating-system documentation.
-y	Causes the database server to automatically respond yes to all prompts	None.
blobspace	Names the blobspace from which the chunk is dropped	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see dropping a chunk from a blobspace, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
dbspace	Names the dbspace from which the chunk is dropped	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see dropping a chunk from a dbspace with **onspaces**, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

Element	Purpose	Key Considerations
sbspace	Names the sbspace from which the chunk is dropped	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For background information, see dropping a chunk from a dbspace with **onspaces**, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.

Important: You must specify a pathname to indicate to the database server that you are dropping a chunk.

Start Mirroring

Start Mirroring:

```
                                  ┌─────────────────────────────────────────────────────┐
                                  ▼                                                     │
├──-m──┬─dbspace───┬────────┬─-p──pathname──-o──offset──-m──pathname──offset─┬──┴────────►
       ├─blobspace─┤        └─-f──filename──────────────────────────────────────┘
       └─sbspace───┘

►──┬────┬───────────────────────────────────────────────────────────────────────────┤
   └─-y─┘
```

Element	Purpose	Key Considerations
-f ***filename***	Indicates that chunk-location information is in a file named *filename*	**Additional Information**: The file must be a buffered file that already exists. The pathname must conform to the operating-system-specific rules for pathnames. **References**: For more information, see "Using a File to Specify Chunk-Location Information with the -f Option" on page 13-23.
-m	Adds mirroring for an existing dbspace, blobspace, or sbspace	**Additional Information**: User-data chunks in a mirrored sbspace need not be mirrored. The mirrored chunks should be on a different disk. You must mirror all the chunks at the same time.
-m ***pathname offset***	The second time that *pathname* occurs in the syntax diagram, it indicates the disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that performs the mirroring. The second time *offset* appears in the syntax diagram, it indicates the offset to reach the mirrored chunk of the newly mirrored dbspace, blobspace, or sbspace. Also see the entries for *pathname* and *offset* in this table.	None.

Element	Purpose	Key Considerations
-o ***offset***	The first time that *offset* occurs in the syntax diagram, it indicates, in kilobytes, the offset into the disk partition or into the unbuffered device to reach the initial chunk of the newly mirrored dbspace, blobspace, or sbspace.	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes. **References**: For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-p ***pathname***	The first time *pathname* occurs in the syntax diagram, it indicates the disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that you want to mirror.	**Additional Information**: The chunk must be an existing unbuffered device or buffered file. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server. **References**: For pathname syntax, see your operating-system documentation.
-y	Causes the database server to automatically respond yes to all prompts	None.
blobspace	Names the blobspace that you want to mirror	**References**: Syntax must conform to the Identifier segment; see *IBM Informix Guide to SQL: Syntax*. For more information, see the chapter on using mirroring in the *IBM Informix Administrator's Guide*.
dbspace	Names the dbspace that you want to mirror	**References**: Syntax must conform to the Identifier segment; see *IBM Informix Guide to SQL: Syntax*. For background information, see the chapter on using mirroring in the *IBM Informix Administrator's Guide*.
sbspace	Names the sbspace that you want to mirror	**References**: Syntax must conform to the Identifier segment; see *IBM Informix Guide to SQL: Syntax*. For background information, see the chapter on using mirroring in the *IBM Informix Administrator's Guide*.

Using a File to Specify Chunk-Location Information with the -f Option

You can create a file that contains the chunk-location information. Then, when you execute **onspaces**, use the **-f** option to indicate to the database server that this information is in a file whose name you specify in ***filename***.

The contents of the file should conform to the following format, with options separated by spaces and each set of primary and mirror chunks on separate lines:

```
primary_chunk_path offset mirror_chunk_path offset
```

If the dbspace that you are mirroring contains multiple chunks, you must specify a mirror chunk for each of the primary chunks in the dbspace that you want to mirror. For an example that enables mirroring for a multichunk dbspace, see starting mirroring for unmirrored dbspaces with **onspaces** in the chapter on using mirroring in the *IBM Informix Administrator's Guide*.

End Mirroring

End Mirroring:

```
├──-r──┬─dbspace───┬──┬──────┬──────────────────────────────────────────┤
       ├─blobspace─┤  └─-y───┘
       └─sbspace───┘
```

Element	Purpose	Key Considerations
-r	Indicates to the database server that mirroring should be ended for an existing dbspace, blobspace, or sbspace	**References**: For background information, see the chapter on using mirroring in the *IBM Informix Administrator's Guide*.
-y	Causes the database server to respond yes to all prompts automatically	None.
blobspace	Names the blobspace for which you want to end mirroring	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see the chapter on using mirroring in the *IBM Informix Administrator's Guide*.
dbspace	Names the dbspace for which you want to end mirroring.	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see the chapter on using mirroring in the *IBM Informix Administrator's Guide*.
sbspace	Names the sbspace for which you want to end mirroring	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For background information, see the chapter on using mirroring in the *IBM Informix Administrator's Guide*.

Change Status of a Mirrored Chunk

Change Chunk Status:

```
├──-s──┬─dbspace───┬──-p──pathname──-o──offset──┬─-D─┬──┬──────┬──────────┤
       ├─blobspace─┤                             └─-O─┘  └─-y───┘
       └─sbspace───┘
```

Element	Purpose	Key Considerations
-D	Indicates that you want to take the chunk down	None.
-o ***offset***	Indicates, in kilobytes, the offset into the disk partition or unbuffered device to reach the chunk	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The offset must be a multiple of the page size. The maximum offset is 4 terabytes. **References**: For more information, see allocating raw disk space on UNIX, in the chapter on managing disk space in the *IBM Informix Administrator's Guide*.
-O	Indicates that you want to restore the chunk and bring it online	None.

Element	Purpose	Key Considerations
-p ***pathname***	Indicates the disk partition or unbuffered device of the chunk	**Additional Information**: The chunk can be an unbuffered device or a buffered file. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server. **References**: For pathname syntax, see your operating-system documentation.
-s	Indicates that you want to change the status of a chunk	**Restrictions**: You can only change the status of a chunk in a mirrored pair. **References**: For more information, see changing the mirror status in the *IBM Informix Administrator's Guide*.
-y	Causes the database server to respond yes to all prompts automatically	None.
blobspace	Names the blobspace whose status you want to change	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see changing the mirror status in the *IBM Informix Administrator's Guide*.
dbspace	Names the dbspace whose status you want to change	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see changing the mirror status in the *IBM Informix Administrator's Guide*.
sbspace	Names the sbspace whose status you want to change	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For background information, see changing the mirror status in the *IBM Informix Administrator's Guide*.

Specify DATASKIP Parameter

Specify DATASKIP:

```
├──-f──┬─OFF─┬──┬───────────────┬──┬────┬──────────────────────────────────┤
       └─ON──┘  └─dbspace-list──┘  └─-y─┘
```

Element	Purpose	Key Considerations
-f	Indicates to the database server that you want to change the DATASKIP default for specified dbspaces or all dbspaces	**Additional Information**: All changes in the DATASKIP status are recorded in the message log.
-y	Causes the database server to automatically respond yes to all prompts	None.
dbspace-list	Specifies the name of one or more dbspaces for which DATASKIP will be turned ON or OFF	**References**: Syntax must conform to the Identifier segment; see the *IBM Informix Guide to SQL: Syntax*. For more information, see "DATASKIP" on page 1-26 and the *IBM Informix Performance Guide*.

Element	Purpose	Key Considerations
OFF	Turns off DATASKIP	**Additional Information**: If you use OFF without *dbspace-list*, DATASKIP is turned off for all fragments. If you use OFF with *dbspace-list*, only the specified fragments are set with DATASKIP off.
ON	Turns on DATASKIP	**Additional Information**: If you use ON without *dbspace-list*, DATASKIP is turned on for all fragments. If you use ON with dbspace-list, only the specified fragments are set with DATASKIP on.

The **onspaces** utility lets you specify DATASKIP on a dbspace level or across all dbspaces.

Chapter 14. The onstat Utility

In This Chapter

The **onstat** utility reads shared-memory structures and provides statistics about the database server at the time that the command executes. The *system-monitoring interface* also provides information about the database server. For information on the system-monitoring interface, see Chapter 2, “The sysmaster Database.”

You can combine multiple **onstat** option flags in a single command. The contents of shared memory might change as the **onstat** output displays. The **onstat** utility does not place any locks on shared memory, so running the utility does not affect performance.

Monitor the Database Server Status

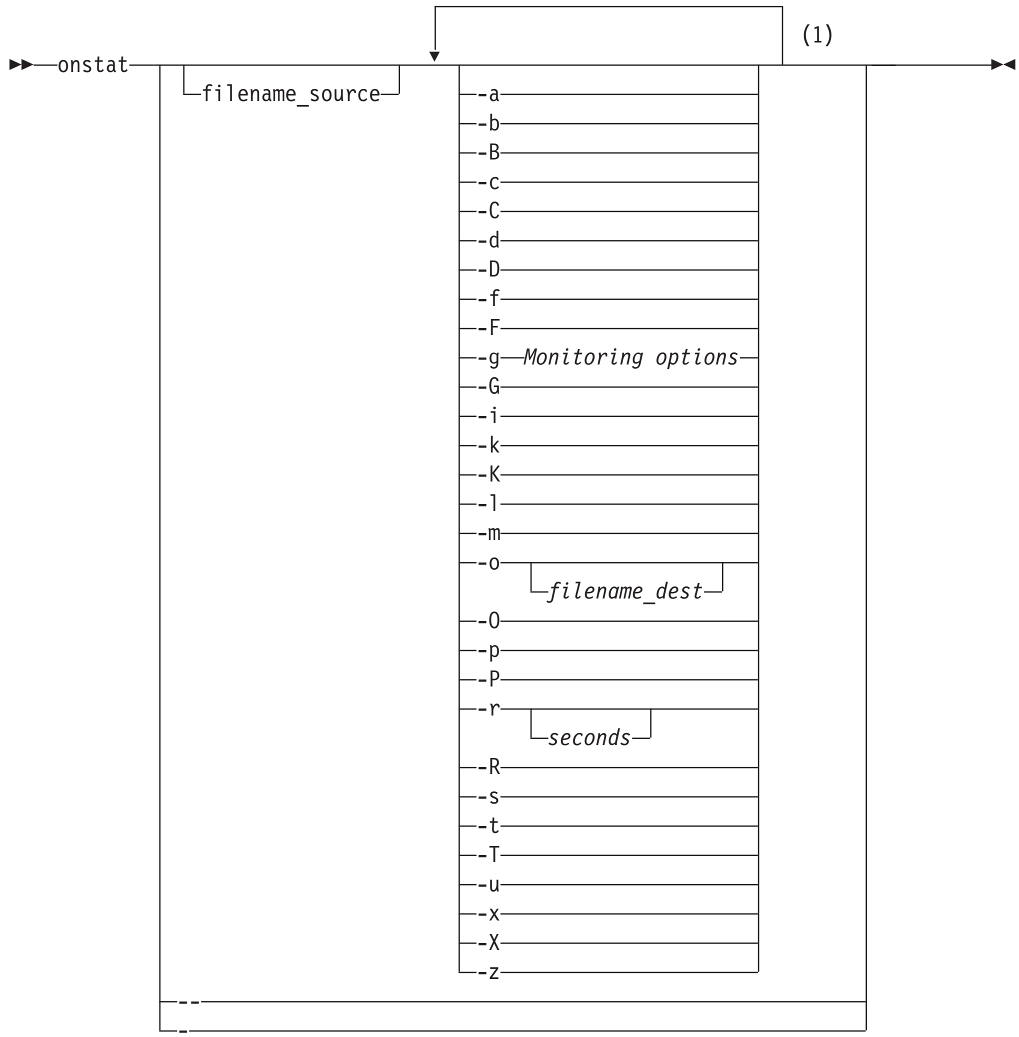
One useful feature of onstat output is the heading that indicates the database server status. Whenever the database server is blocked, **onstat** displays the following line after the banner line:

```
Blocked: reason
```

The variable *reason* can take one of the following values.

Reason	Description
CKPT	Checkpoint
LONGTX	Long transaction
ARCHIVE	Ongoing archive
MEDIA_FAILURE	Media failure
HANG_SYSTEM	Database server failure
DBS_DROP	Dropping a dbspace
DDR	Discrete high-availability data replication
LBU	Logs full high-watermark

Syntax



Notes:

1 Only one occurrence of each item is allowed. More than one option can be specified on a single onstat command invocation.

Element	Purpose	Key Considerations
-	Displays the output header	References: See "Output Header" on page 14-7.
--	Displays a listing of all **onstat** options and their functions	**Additional Information**: This option is the only option flag that you cannot combine with any other flag. **References:** See "onstat --" on page 14-8.
-a	Interpreted as **onstat -cuskbtdlp**. Displays output in that order	**References:** See "onstat -a" on page 14-8.
-b	Displays information about buffers currently in use, including number of resident pages in the buffer pool	**References:** See "onstat -b" on page 14-8.
-B	Obtains information about all database server buffers, not just buffers currently in use. See the entry for **-b** in this table	**Additional Information**: The **-B** output display fields are the same as the fields that appear in the **-b** output.

Element	Purpose	Key Considerations
-c	Displays the ONCONFIG file: • **$INFORMIXDIR/etc/ $ONCONFIG** for UNIX • **%INFORMIXDIR%\etc\ %ONCONFIG%** for Windows	**References:** See "onstat -c" on page 14-10.
-C	Prints B-tree scanner information	**References:** See " "onstat -C" on page 14-10.
-d	Displays information for chunks in each storage space	**References:** See "onstat -d" on page 14-11.
-D	Displays page-read and page-write information for the first 50 chunks in each dbspace	**References:** See "onstat -D" on page 14-14.
-f	Lists the dbspaces currently affected by the DATASKIP feature	**References:** See "onstat -f" on page 14-15.
-F	Displays a count for each type of write that flushes pages to disk	**References:** See "onstat -F" on page 14-15.
-g	Provides monitoring options	**References:** See "onstat -g Monitoring Options" on page 14-16.
-G	Prints global transaction IDs	**References**: See "onstat -G" on page 14-77.
-i	Puts the **onstat** utility into interactive mode	**References:** See "onstat -i" on page 14-78.
-j	Prints the interactive status of the active **onpload** process	**References** : See "onstat -j" on page 14-79.
-k	Displays information about active locks	**References:** See "onstat -k" on page 14-80.
-l	Displays information about physical and logical logs, including page addresses	**References:** See "onstat -l" on page 14-82.
-m	Displays the 20 most recent lines of the database server message log	**Additional Information**: Output from this option lists the full pathname of the message-log file and the 20 file entries. A date-and-time header separates the entries for each day. A time stamp prefaces single entries within each day. The name of the message log is specified as MSGPATH in the ONCONFIG file. **References:** See "onstat -m" on page 14-84.
-o	Saves copies of the shared-memory segments to ***filename***	**Additional Information**: If you omit a filename in the **onstat** command, the copy of shared memory is saved to **onstat.out** in the current directory.
-O	Displays information about the Optical Subsystem memory cache and staging-area blobspace	**References:** See "onstat -O" on page 14-85.
-p	Displays profile counts.	**References:** See "onstat -p" on page 14-86.
-P	Displays for all partitions the partition number and the break-up of the buffer-pool pages that belong to the partition	**References:** See "onstat -P" on page 14-89.
-r	Repeats the accompanying **onstat** options after they wait the specified *seconds* between each execution. The default value of *seconds* is 5.	**Additional Information**: To end execution, press `DEL` or `CTRL-C`.

Element	Purpose	Key Considerations
-R	Displays detailed information about the LRU queues, FLRU queues, and MLRU queues	**References:** See "onstat -R" on page 14-90.
-s	Displays general latch information	**References:** See "onstat -s" on page 14-92.
-t	Displays tblspace information, including residency state, for active tblspaces	**References:** See "onstat -t and -T" on page 14-94.
-T	Displays tblspace information for all tblspaces	**References:** See "onstat -t and -T" on page 14-94.
-u	Prints a profile of user activity	**References:** See "onstat -u" on page 14-95.
-x	Displays information about transactions	**References:** See "onstat -x" on page 14-97.
-X	Obtains precise information about the threads that are sharing and waiting for buffers	**References:** See "onstat -X" on page 14-99.
-z	Sets the profile counts to 0	**References:** See "onstat -z" on page 14-101.
filename_dest	Specifies destination file for the copy of the shared-memory segments	**Restrictions**: Name must not match the name of any existing file. **References**: For pathname syntax, see your operating-system documentation.
filename_source	Specifies file that **onstat** reads as source for the requested information	**Restrictions**: This file must include a previously stored shared-memory segment that you created with the **-o** option of **onstat**. **References**: For specific details on this option, see "Statistics Culled from Source File." For pathname syntax, see your operating-system documentation.
Monitoring options	Specifies which **onstat -g** monitoring option to use	**References:** See "onstat -g Monitoring Options" on page 14-16.
seconds	Specifies number of seconds between each execution of the **onstat -r** command	**Restrictions**: This value must be an unsigned integer greater than 0.

Statistics Culled from Source File

Use the *filename_source* parameter with other option flags to derive the requested **onstat** statistics from the shared-memory segments that *filename_source* contains. You must first use the **onstat -o** command to create a file that contains the shared-memory segments.

Interactive Execution

To put the **onstat** utility in interactive mode, use the **-i** option. Interactive mode allows you to enter multiple options, one after the other, without exiting the program. For information on using interactive mode, see "onstat -i" on page 14-78.

Continuous onstat Execution

Use the *seconds* parameter with the **-r** option flag to cause all other flags to execute repeatedly after they wait the specified seconds between each execution.

Output Header

All **onstat** output includes a header. The **onstat -** option displays only the output header and is useful for checking the database server mode. The header takes the following form:

`Version--Mode (Type)--(Checkpnt)--Up Uptime--Sh_mem Kbytes`

Version
: Is the product name and version number

Mode
: Is the current operating mode.

(Type)
: If the database server uses High-Availability Data Replication, indicates whether the type is primary or secondary

  If the database server is not involved in data replication, this field does not appear. If the type is primary, the value `P` appears. If the type is secondary, the value `S` appears.

(Checkpnt)
: Is a checkpoint flag

  If it is set, the header might display two other fields after the mode if the timing is appropriate:

(CKPT REQ)	Indicates that a user thread has requested a checkpoint
(CKPT INP)	Indicates that a checkpoint is in progress. During the checkpoint, access is limited to read only. The database server cannot write or update data until the checkpoint ends

Uptime
: Indicates how long the database server has been running

Sh_mem
: Is the size of database server shared memory, expressed in kilobytes

A sample header for the database server follows:

```
Dynamic Server Version 10.00.UC1--On-Line--Up 15:11:41--9216 Kbytes
```

Logs Full Subheader

If the database server is blocked, the **onstat** header output includes an extra line that reads as follows:

`Blocked: reason(s)`

The reason can be one or more of the following.

Reason	Explanation
CKPT	Checkpoint
LONGTX	Long transaction
ARCHIVE	Ongoing storage-space backup
MEDIA_FAILURE	Media failure
HANG_SYSTEM	Database server failure
DBS_DROP	Dropping a dbspace
DDR	Discrete data replication

LBU	Logs full high-watermark

onstat

If you invoke **onstat** without any options, the command is interpreted as **onstat -pu** (**-p** option and **-u** option).

onstat --

The **--** option displays a listing of all **onstat** options and their functions. This option is the only option flag that you cannot combine with any other flag.

onstat -a

The **-a** option is interpreted as **onstat -cuskbtdlp**, and output is displayed in that order. For an explanation of each option, refer to the appropriate flag in the paragraphs that follow.

onstat -b

The **-b** option displays information about buffers currently in use, including the total number of resident pages in the buffer pool. (For information about all buffers, not just those in use, use **onstat -B**.)

The maximum number of buffers available is specified in the **buffers** field in the BUFFERPOOL configuration parameter in the ONCONFIG file.

The **-b** and **-B** options also provide summary information about the number of modified buffers, the total number of resident pages in the buffer pool, the total number of buffers available, the number of hash buckets available, and the size of the buffer in bytes (the page size).

```
123 modified, 23 resident, 2000 total, 2048 hash buckets, 2048 buffer size.
```

Example Output:

```
   IBM Informix Dynamic Server Version 10.00.UC1       -- On-Line -- Up 18:35:04
-- 34816 Kbytes
   Buffers
   address  userthread flgs pagenum       memaddr  nslots pgflgs xflgs owner   waitlist

   Buffer pool page size: 2048
    0 modified, 3000 total, 4096 hash buckets, 2048 buffer size

   Buffer pool page size: 8192
    0 modified, 1000 total, 1024 hash buckets, 8192 buffer size
```

Figure 14-1. onstat -B Output

Output Description:

You can interpret output from the **-b** and the **-B** options as follows:

Buffer pool page size
: is the size of the buffer pool pages in bytes

address
: Is the address of the buffer header in the buffer table

userthread
: is the address of the most recent user thread to access the buffer table. Many user threads might be reading the same buffer concurrently.

flgs
: Uses the following flag bits to describe the buffer:

0x01	Modified data
0x02	Data
0x04	LRU
0x08	Error

pagenum
: Is the physical page number on the disk

memaddr
: Is the buffer memory address

nslots
: Is the number of slot-table entries in the page

This field indicates the number of rows (or portions of a row) that are stored on the page.

pgflgs
: Uses the following values, alone or in combination, to describe the page type:

1	Data page
2	Tblspace page
4	Free-list page
8	Chunk free-list page
9	Remainder data page
b	Partition resident blobpage
c	Blobspace resident blobpage
d	Blob chunk free-list bit page
e	Blob chunk blob map page
10	B-tree node page
20	B-tree root-node page
40	B-tree branch-node page
80	B-tree leaf-node page
100	Logical-log page
200	Last page of logical log
400	Sync page of logical log
800	Physical log
1000	Reserved root page
2000	No physical log required
8000	B-tree leaf with default flags

xflgs
: Uses the following flag bits to describe buffer access:

0x10	share lock
0x80	exclusive lock

owner
: Is the user thread that set the **xflgs** buffer flag

waitlist Is the address of the first user thread that is waiting for access to this buffer

For a complete list of all threads waiting for the buffer, refer to “onstat -X” on page 14-99.

onstat -c

Use the **onstat -c** option to display the contents of the ONCONFIG file. The database server first checks if you have assigned a value to the environment variable **ONCONFIG**. You can use the **onstat -c** option with the database server in any mode, including offline.

UNIX Only

On UNIX, if you have set **ONCONFIG**, **onstat -c** displays the contents of the **$INFORMIXDIR/etc/$ONCONFIG** file. If not, by default, **onstat -c** displays the contents of **$INFORMIXDIR/etc/onconfig**.

End of UNIX Only

Windows Only

On Windows, if you have set **ONCONFIG**, **onstat -c** displays the contents of the **%INFORMIXDIR%\etc\%ONCONFIG%** file. If not, by default, **onstat -c** displays the contents of **%INFORMIXDIR%\etc\onconfig**.

End of Windows Only

onstat -C

Use the **-C** option to print the file information about the B-tree scanner subsystem and each B-tree scanner thread. The following options are available with the **onstat -C** command:

prof Prints the profile information for the system and each B-tree scanner thread

hot Prints the hot list index key in the order to be cleaned

part Prints all partitions with index statistics

clean Prints information about all the partitions that were cleaned or need to be cleaned.

range Prints the savings in pages processes by using index range scanning

all Prints all **onstat -C** options

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1       -- On-Line -- Up 03:59:17 -- 15360
Kbytes

Btree Cleaner Info
BT scanner profile Information
==============================
Active Threads                              1
Global Commands                         20000   Building hot list
Number of partition scans                   0
Main Block                         0x0a69cc08
BTC Admin                          0x0a4d9248


BTS info     id   Prio    Partnum      Key     Cmd
0xa69cd58     0    Low   0x00000000     0       40  Yield N
   Number of leaves pages scanned                        0
   Number of leaves with deleted items                   0
   Time spent cleaning (sec)                             0
   Number of index compresses                            0
   Number of deleted items                               0
   Number of index range scans                           0
   Number of index leaf scans                            0
```

Figure 14-2. onstat -C Output

onstat -d

Use the **-d** option to display information for chunks in each storage space. You can interpret output from this option as follows.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1     -- On-Line -- Up 00:01:23 -- 27648 Kbytes

Dbspaces
address  number   flags      fchunk   nchunks  pgsize   flags    owner    name
a3217d8  1        0x60001    1        1        2048     N  B     informix rootdbs
a426e40  2        0x60001    2        1        4096     N  B     informix dbsp1
a415630  3        0x60001    3        1        8192     N  B     informix dbsp2
 3 active, 2047 maximum

Chunks
address  chunk/dbs  offset     size       free       bpages     flags pathname
a321928  1     1    0          30000      12173                 PO-B  /local1/engines
a321b00  2     2    0          2400       2347                  PO-B  /local1/engines
a415780  3     3    0          1200       1147                  PO-B  /local1/engines
 3 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.

Expanded chunk capacity mode: always
```

Figure 14-3. onstat -d Output

Output Description:

The first section of the display describes the storage spaces:

address
: Is the address of the storage space in the shared-memory space table

number
: Is the unique ID number of the storage space assigned at creation

flags
: Uses the following hexadecimal values to describe each storage space:

0x00000000	Mirror not allowed and dbspace is unmirrored
0x00000001	Mirror is allowed and dbspace is unmirrored
0x00000002	Mirror is allowed and dbspace is mirrored
0x00000004	Down
0x00000008	Newly mirrored
0x00000010	Blobspace
0x00000020	Blobspace on removable media
0x00000040	Blobspace is on optical media
0x00000080	Blobspace is dropped
0x00000100	Blobspace is the optical STAGEBLOB
0x00000200	Space is being recovered
0x00000400	Space is fully recovered
0x00000800	Logical log is being recovered
0x00001000	Table in dbspace is dropped
0x00002000	Temporary dbspace
0x00004000	Blobspace is being backed up
0x00008000	Sbspace
0x0000a001	Temporary sbspace
0x00010000	Physical or logical log changed
0x00020000	Dbspace or chunk tables have changed
0x20002	Dbspace or chunk tables have changed and dbspace is mirrored
0x60001	Dbspace has large chunks and is unmirrored. Any changes triggers changes on rootdbspace

fchunk
: Is the ID number of the first chunk

nchunks
: Is the number of chunks in the storage space

pgsize
: Is the size of the dbspace pages in bytes

flags
: Uses the following letter codes to describe each storage space:

Position 1:

M	Mirrored
N	Not mirrored

Position 2:

X	Newly mirrored

P Physically recovered, waiting for P -- logical recovery

L Being logically recovered

R Being recovered

Position 3:

B Blobspace

S Sbspace

Position 4:

B Dbspace has large chunks greater than 2 GB

owner Is the owner of the storage space

name Is the name of the storage space

In the line immediately following the storage-space list, **active** refers to the current number of storage spaces in the database server instance including the rootdbs and **maximum** refers to total *allowable* spaces for this database server instance.

The second section of the **onstat -d** output describes the chunks:

address Is the address of the chunk

chk/dbs Is the chunk number and the associated space number

offset Is the offset into the file or raw device in pages

size Is the size of the chunk in terms of the page size of the dbspace to which it belongs.

free Is the number of free pages in the chunk in terms of the page size of the dbspace to which it belongs.

For a blobspace, a tilde indicates an approximate number of free blobpages.

For an sbspace, indicates the number of free pages of user data space and total user data space.

bpages Is the size of the chunk in blobpages

Blobpages can be larger than disk pages; therefore, the **bpages** value can be less than the **size** value.

For an sbspace, is the size of the chunk in sbpages

flags Provides the chunk status information as follows:

Position 1:

P Primary

M Mirror

Position 2:

N Renamed and either Down or Inconsistent

O Online

D Down

X Newly mirrored

I Inconsistent

Position 3:

- \- Dbspace
- B Blobspace
- S Sbspace
- T Temporary dbspace

Position 4:

- B Has large chunks greater than 2 GB

pathname Is the pathname of the physical device

In the line immediately following the chunk list, **active** displays the number of active chunks (including the root chunk) and **maximum** displays the total number of chunks.

For information about page reads and page writes, refer to "onstat -D."

Using onstat -d with Sbspaces

For information about using **onstat -d** to determine the size of sbspaces, user-data areas, and metadata areas, see monitoring sbspaces in the *IBM Informix Administrator's Guide*.

Using onstat -d with Blobspaces

If you issue the **onstat -d** command on an instance with blobspace chunks, the database server displays the following message:

```
NOTE: For BLOB chunks, the number of free pages shown is out of date.
 Run 'onstat -d update' for current stats.
```

To obtain the current statistics for blobspace chunks, issue the **onstat -d update** command. The **onstat** utility updates shared memory with an accurate count of free pages for each blobspace chunk. The database server displays the following message:

```
Waiting for server to update BLOB chunk statistics ...
```

onstat -D

Use the **-D** option to display page-read and page-write information for the first 50 chunks in each space.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1    -- On-Line -- Up 03:59:42 -- 34816 Kbytes

Dbspaces
address  number   flags      fchunk   nchunks pgsize  flags    owner    name
a40d7d8  1        0x1        1        1       2048    N        informix rootdbs
 1 active, 2047 maximum

Chunks
address  chunk/dbs  offset     page Rd  page Wr  pathname
a40d928  1      1   0          0        0        /work/10.0/dbspaces/stardbs3
 1 active, 2047 maximum

Expanded chunk capacity mode: disabled
```

Figure 14-4. onstat -D Output

Output Description:

The output of **onstat -D** is almost identical to the output of **onstat -d**. The following columns are unique to **onstat -D**. For information on the other output columns see "onstat -d" on page 14-11.

page Rd Is the number of pages read

page Wr Is the number of pages written

onstat -f

Use the **-f** option to list the dbspaces that the dataskip feature currently affects. The **-f** option lists both the dbspaces that were set with the DATASKIP configuration parameter and the **-f** option of **onspaces**. When you execute **onstat -f**, the database server displays one of the following three outputs:

- Dataskip is OFF for all dbspaces.
- Dataskip is ON for all dbspaces.
- Dataskip is ON for the following dbspaces:

  `dbspace1 dbspace2...`

onstat -F

Use the **-F** option to display a count for each type of write that flushes pages to disk.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 04:00:17 -- 15360 Kbytes


Fg Writes     LRU Writes    Chunk Writes
0             0             0

address  flusher  state    data
a4d8628  0        I        0       = 0X0
      states: Exit Idle Chunk Lru
```

Figure 14-5. onstat -F Output

Output Description:

You can interpret output from this option as follows:

Fg Writes — Is the number of times that a foreground write occurred

LRU *Writes* — Is the number of times that an LRU write occurred

Chunk Writes — Is the number of times that a chunk write occurred

address — Is the address of the user structure assigned to this page-cleaner thread

flusher — Is the page-cleaner number

state — Uses the following codes to indicate the current page-cleaner activity:

- C — Chunk write
- E — Exit
- I — Cleaner is idle
- L — LRU queue

The exit code indicates either that the database server is performing a shutdown or that a page cleaner did not return from its write in a specific amount of time. When an operation fails to complete within the allotted time, this situation is known as a time-out condition. The database server does not know what happened to the cleaner, so it is marked as `exit`. In either case, the cleaner thread eventually exits.

data — Provides additional information in concert with the **state** field

If **state** is `C`, **data** is the chunk number to which the page cleaner is writing buffers. If **state** is `L`, **data** is the LRU queue from which the page cleaner is writing. The **data** value is displayed as a decimal, followed by an equal sign, and repeated as a hexadecimal.

onstat -g Monitoring Options

The following **onstat -g** options are provided for support and debugging only. You can include only one of these options per **onstat -g** command. For more information, see your *IBM Informix Performance Guide*.

onstat -g Option	Topic or Function
-g act	Prints active threads. For example output, see "The onstat -g act Option" on page 14-21.
-g afr ***pool name | session id***	Prints allocated memory fragments for a specified session or shared-memory pool. Each session is allocated a pool of shared memory. To obtain the pool name, see the **-mem** option. For example output, see "The onstat -g afr pool name | session id Option" on page 14-22.
-g all	Prints all multithreading information.
-g ath	Prints all threads. The **sqlmain** threads represent client sessions. The **rstcb** value corresponds to the **user** field of the **onstat -u** command. For example output, see "The onstat -g ath Option" on page 14-22. For information on using **onstat -g ath** to print Enterprise Replication threads, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

onstat -g Option	Topic or Function
-g cat [*modifier*]	Prints information from the Enterprise Replication global catalog. The global catalog contains a summary of information about the defined servers, replicates, and replicate sets on each of the servers within the enterprise. For more information and sample output, see "The onstat -g cat Option" on page 14-23.
-g cac agg	Prints the definitions for user-defined aggregates that are currently in the cache.
-g cac stmt	Prints the contents of the SQL statement cache. Prints the same output as the **-g ssc** statement.
-g con	Prints conditions with waiters. For example output, see "The onstat -g con Option" on page 14-24.
-g ddr	Prints the status of the Enterprise Replication database log reader. If log reading is blocked, data might not be replicated until the problem is resolved. For more information and sample output, see "The onstat -g ddr Option" on page 14-25.
-g dic ***table***	Without any parameters, prints one line of information for each table cached in the shared-memory dictionary. If given a specific table name as a parameter, prints internal SQL information for that table. For more information, see your *IBM Informix Performance Guide*. For sample output, see "The onstat -g dic table Option" on page 14-26.
-g dis	Prints a list of database servers and their status, and information about each database server, **INFORMIXDIR**, **sqlhosts** file, ONCONFIG file, and hostname. For example output, see "The onstat -g dis Option" on page 14-27.
-g dll	Prints a list of dynamic libraries that have been loaded. For example output, see "The onstat -g dis Option" on page 14-27.
-g dri	Prints data-replication information. See monitoring High-Availability Data-Replication status in the *IBM Informix Administrator's Guide*. For example output, see "The onstat -g dri Option" on page 14-28.
-g dsc	Prints data-distribution cache information. For example output, see "The onstat -g dsc Option" on page 14-29.
-g dss [*modifier*]	Prints detailed statistical information about the activity of individual data sync threads and about user-defined data types. For more information and sample output, see "The onstat -g dss Option" on page 14-29.
-g dtc	Prints statistics about the delete table cleaner which removes rows from the delete table when they are no longer needed. For more information and sample output, see "The onstat -g dtc Option" on page 14-30.
-g env	Prints the values of environment variables the database server currently uses. For more information, see "The onstat -g env Option" on page 14-31.
-g ffr ***pool name | session id***	Prints free fragments for a pool of shared memory. For example output, see "The onstat -g ffr pool name | session id Option" on page 14-33.

onstat -g Option	Topic or Function
-g glo	Prints global multithreading information. This information includes CPU use information about the virtual processors, the total number of sessions, and other multithreading global counters. On Windows, the virtual processors are operating system threads. The values displayed under the 'pid' field are thread ids not process ids. (Windows). For example output, see "The onstat -g glo Option" on page 14-33.
-g grp [*modifier*]	Prints statistics about the Enterprise Replication grouper. The grouper evaluates the log records, rebuilds the individual log records into the original transaction, packages the transaction, and queues the transaction for transmission. For more information and sample output, see "The onstat -g grp Option" on page 14-35.
-g imc	Prints information about MaxConnect instances that are connected to the database server. If MaxConnect is not connected to the database server, this command displays No MaxConnect servers are connected.
-g ioa	Prints combined information from **-g ioq** and **-g iov**. For example output, see "The onstat -g ioa Option" on page 14-40.
-g iob	Prints the big buffer usage summary. For example output, see "The onstat -g iob Option" on page 14-41.
-g iof	Prints asynchronous I/O statistics by chunk or file. This option is similar to the **-D** option, except it also displays information on nonchunk, temporary, and sort-work files. For example output, see "The onstat -g iof Option" on page 14-42.
-g iog	Prints AIO global information. For example output, see "The onstat -g iog Option" on page 14-42.
-g ioq ***queue name***	Prints pending I/O operations for the *queue name*. If given the *gfd* or *kaio* queue name, a queue for each CPU VP is displayed. If *queue name* is omitted, I/O statistics for all queues are displayed. For example output, see "The onstat -g ioq queue name Option" on page 14-43.
-g iov	Prints asynchronous I/O statistics by virtual processor. For example output, see "The onstat -g iov Option" on page 14-44.
-g lmx	Prints all locked mutexes. For example output, see "The onstat -g lmx Option" on page 14-45.
-g lsc	Displays information about light scans.
-g mem ***pool name*** **|** ***session id***	Prints statistics for a memory pool. Also displays the pool name, type of shared memory segment that contains the pool, the address of the pool, the total size of the pool, the number of bytes of free memory that it contains, and the number of free and allocated fragments in the pool. If no argument is provided, displays information about all pools. The block pools are listed in a separate section after the main pool list. You also can use ISA to obtain detailed information about a memory pool. If you run an SQL query that allocates memory from the PER_STMT_EXEC and PER_STMT_PREP memory duration pools, **onstat -g mem** displays information on the **PRP.sessionid.threadid** pool and the **EXE.sessionid.threadid** pool. For example output, see "The onstat -g mem pool name | session id Option" on page 14-46. For more information, see the *IBM Informix DataBlade API Programmer's Guide*.
-g mgm	Prints Memory Grant Manager resource information. For example output, see "The onstat -g mgm Option" on page 14-47.

onstat -g Option	Topic or Function
-g nbm	Prints block bit map for the nonresident segments, one bit per 8-kilobyte block. Bit set indicates block free. For example output, see "The onstat -g nbm Option" on page 14-49.
-g nif [*modifier*]	Prints statistics about the network interface. Useful to determine why data is not replicating. For more information and sample output, see "The onstat -g nif Option" on page 14-50.
-g nsc *client id*	Prints shared-memory status by *client id*. If *client id* is omitted, all client status areas are displayed. This command prints the same status data as the **nss** command. For example output, see "The onstat -g nsc client_id Option" on page 14-51.
-g nsd	Prints network shared-memory data for poll threads. For example output, see "The onstat -g nsd Option" on page 14-53.
-g nss *session id*	Prints network shared-memory status by ***session id***. If ***session id*** is omitted, all session status areas are displayed. This command prints the same status data as the **nsc** command.
-g nta	Prints combined network statistics from **-g ntd**, **-g ntm**, **-g ntt**, and **-g ntu**. If MaxConnect is installed, this command prints statistics that you can use to tune MaxConnect performance.
-g ntd	Prints network statistics by service. For example output, see "The onstat -g ntd Option" on page 14-54.
-g ntm	Prints network mail statistics. For example output, see "The onstat -g ntm Option" on page 14-54.
-g ntt	Prints network user times. For example output, see "The onstat -g ntt Option" on page 14-55.
-g ntu	Prints network user statistics. For example output, see "The onstat -g ntu Option" on page 14-55.
-g pos	Prints **$INFORMIXDIR/etc/ .infos.DBSERVERNAME fil**e for UNIX and **%INFORMIXDIR%\etc\ .infos.DBSERVERNAME** for Windows. For example output, see "The onstat -g pos Option" on page 14-55.
-g ppf *partition number* | 0	Prints partition profile for ***partition number***; 0 prints profiles for all partitions. If TBLSPACE_STATS configuration parameter is set to 0, displays: Partition profiles is disabled. For example output, see "The onstat -g ppf partition number | 0 Option" on page 14-56.
-g prc	Prints information about SPL routine cache. For example output, see "The onstat -g prc Option" on page 14-57.
-g qst	Prints queue statistics.
-g que	Prints statistics for the high-level queue interface (which are common to all the queues of the Enterprise Replication Queue Manager. For more information and sample output, see "The onstat -g que Option" on page 14-57.
-g rbm	Prints block bit map for the resident segment (communication message area). For example output, see "The onstat -g rbm Option" on page 14-58.
-g rcv [*serverid*]	Prints statistics about the receive manager, which is a set of service routines between the receive queues and data sync. For more information and sample output, see "The onstat -g rcv Option" on page 14-59.
-g rea	Prints ready threads. For example output, see "The onstat -g rea Option" on page 14-62.

onstat -g Option	Topic or Function
-g rep [*replname*]	Prints events that are in the queue for the schedule manager. For more information and sample output, see "The onstat -g rep Option" on page 14-62.
-g rqm [*modifier*]	Prints statistics and contents of the low-level queues (each individual queue) managed by the Reliable Queue Manager (RQM). For more information and sample output, see "The onstat -g rqm Option" on page 14-62.
-g rwm	Prints read/write mutexes. For example output, see "The onstat -g rwm Option" on page 14-65.
-g sch	Prints the number of semaphore operations, spins, and busy waits for each virtual processor. On Windows, the virtual processors are operating system threads. The values displayed under the 'pid' field are thread ids not process ids. (Windows) For example output, see "The onstat -g sch Option" on page 14-66.
-g seg	Prints shared-memory-segment statistics. This option shows the number and size of shared-memory segments that the database server is currently using. For example output, see "The onstat -g seg Option" on page 14-66.
-g ses *sessionid*	Prints session information by ***sessionid***. If ***sessionid*** is missing, a one-line summary of each session prints. For more information, see "The onstat -g ses Option" on page 14-67.
-g sle	Prints all sleeping threads. For example output, see "The onstat -g sle Option" on page 14-71.
-g smb *option*	Prints detailed information about sbspaces: • **c** = lists all the chunks in the sbspace. • **fdd** = lists the smart-large-object file descriptors. • **lod** = lists the smart-large-object headers in the header table. • **s** = lists the sbspace attributes (owner, name, page size, **-Df** flag settings). Fields with a value of 0 or -1 were not initialized during sbspace creation.
-g spi	Prints spin locks that virtual processors have spun more than 10,000 times to acquire. These spin locks are called *longspins*. The total number of longspins is printed in the heading of the **glo** command. Excessive longspins might indicate an overloaded system, too many virtual processors for a given computer or node, or an internal problem. To reduce longspins, reduce the number of virtual processors (generally class CPU), reduce the load on the computer, or use the *no-age* or *processor* affinity features.
-g sql *session id*	Prints SQL information by *session id*. If *session id* is omitted, a one-line summary for each session prints. For more information, see "The onstat -g sql Option" on page 14-71.
-g ssc	Monitors the number of times that the database server reads the SQL statement in the cache. For example output, see "The onstat -g ssc Option" on page 14-72 Displays the same output as **onstat -g cac stmt**. For more information, see improving query performance in the *IBM Informix Performance Guide*.

onstat -g Option	Topic or Function
-g ssc all	Reports the *key-only* cache entries as well as the fully cached statements. If the value in the **hits** column is less than the STMT_CACHE_HITS value, that entry is a *key-only* cache entry. For more information, see memory utilization in the *IBM Informix Performance Guide*.
-g ssc pool	Reports usage of all memory pools for the SQL statement cache. The output displays information on the name, class, address, and total size of the memory pools. For more information, see improving query performance in the *IBM Informix Performance Guide*.
-g stk ***tid*** | **all**	Dumps stack of thread specified by thread ID or stacks for *all* threads. This option is not supported on all platforms and is not always accurate. For example output, see "The onstat -g stk tid Option" on page 14-74.
-g stm [***session id***]	Displays the memory that each prepared SQL statement uses. For example output, see "The onstat -g stm Option" on page 14-74. For more information, see memory utilization and improving query performance in the *IBM Informix Performance Guide*.
-g sts	Prints maximum and current stack use per thread. For example output, see "The onstat -g sts Option" on page 14-74.
4 4 **-g sync**	Shows which sync is active. For example output, see "The onstat -g sync Option" on page 14-75.
-g tpf ***tid***	Prints thread profile for a specific thread ID. For example output, see "The onstat -g tpf tid Option" on page 14-76.
-g ufr ***pool name*** | ***session id***	Prints allocated fragments by use.
-g wai	Prints waiting threads; all threads waiting on mutex or condition, or yielding.
-g wmx	Prints all mutexes with waiters. For example output, see "The onstat -g wmx Option" on page 14-77.
-g wst	Prints wait statistics.

The onstat -g act Option

The **onstat -g act** option prints active threads.

Following is sample output from the **onstat -g act** command. For a description of the output, see "The onstat -g ath Option" on page 14-22.

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 18:47:42
-- 101376 Kbytes
Running threads:
tid   tcb        rstcb  prty  status    vp-class      #scheds name
*2    b3132d8    0      2     running   2adm          0       adminthd
*40   c5384d0    0      2     running   1cpu         102630   tlitcppoll
```

Figure 14-6. onstat -g act Output

The onstat -g afr pool name | session id Option

The **onstat -g afr** option prints allocated memory fragments for a specified session or shared-memory pool. Each session is allocated a pool of shared memory.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 18:47:42
-- 43008 Kbytes
Allocations for pool name dfm_pool:
addr            size       memid
10ac8c000       192        overhead
10ac8d000       24352      dfm
```

Figure 14-7. onstat -g afr Output

Output Description:

addr Memory address of the pool fragment

size Size, in bytes, of the pool fragment

memid Memory ID of the pool fragment

The onstat -g ath Option

The **onstat -g ath** option prints information about all threads.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 18:47:42
-- 101376 Kbytes
Threads:
tid    tcb       rstcb    prty  status             vp-class   #scheds name
*2     b3132d8   0        2     running            2adm       0       adminthd
3      b313650   0        2     sleeping forever   3cpu       0       childthd
*5     b4c4028   0        4     sleeping secs: 0   1cpu       9960    Cosvr
 6     b4de2d0   0        2     sleeping secs: 1   3cpu       2014    dfm_svc
*7     b4de640   0        4     sleeping forever   1cpu       91469   xmf_svc
...
*49    c58a450   b3b13a8 4     sleeping secs: 1   1cpu       2014    onmode_mon
1511   c8277c8   b3b1a68  3     cond wait netnorm  5cpu      0      onbar_2
1512   c85b378   b3b2128  2     sleeping forever   5cpu      0      x_exec_1.61
1515   c5f14f0   b3b27e8  2     sleeping forever   1cpu      0      llbu_2
```

Figure 14-8. onstat -g ath Output

Output Description:

tid Thread ID
tcb Thread control block access
rstcb RSAM thread control block access
prty Thread priority
status Thread status
vp-class Virtual processor class
#scheds name Thread name

The onstat -g cat Option

The **onstat -g cat** command prints information from the Enterprise Replication global catalog. The global catalog contains a summary of information about the defined servers, replicates, and replicate sets on each of the servers within the enterprise. If a replicated table is undergoing an alter operation, the **onstat -g cat** command shows that it is in alter mode. For example, use this command to determine:

- How many servers and how many replicates are configured
- Which table matches a given replicate
- Whether a server is a root or leaf server
- The current bitmap mask for a given server. You can use the bitmap mask with the output from the **onstat -g rqm** command to determine which server Enterprise Replication is waiting on for an acknowledgement.

The **onstat -g cat** command has the following formats:

```
onstat -g cat
onstat -g cat scope
onstat -g cat replname
```

The following table describes *replname* and *scope*.

Modifier	Description
replname	The name of a replicate
scope	One of the following values: **servers**—Print information on servers only **repls**—Print information on replicates only **full**—Print expanded information for both replicate servers and replicates

Example Output:

This sample output from the **onstat -g cat repls** command shows that the table **tab** is in alter mode. The replicate **rep1** is defined on this table, its replicate ID is 6553601. For more information on the replicate attributes that this command displays, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 00:01:39 -- 28672 Kbytes
GLOBAL-CATALOG CACHE STATISTICS
REPLICATES
-------------------
Parsed statements:
    Id 6553601 table tab
    Id 6553602 table tab12
Inuse databases: test(2)
  Name: rep1, Id: 6553601 State: ACTIVE Flags: 0x800000 ALTERMODE
          use 0 lastexec Wed Dec 31 18:00:00 1969
    Local Participant: test:nagaraju.tab
    Attributes: TXN scope, Enable ATS, Enable RIS, all columns
          sent in updates
    Conflict resolution: [TIMESTAMP]
    Column Mapping: ON, columns INORDER, offset 8, uncomp_len 12
    Column Name Verifcation: ON
    No Replicated UDT Columns
  Name: rep12, Id: 6553602 State: ACTIVE Flags: 0x800000 use 0
          lastexec Wed Dec 31 18:00:00 1969
    Local Participant: test:nagaraju.tab12
    Attributes: TXN scope, Enable ATS, Enable RIS, all columns
          sent in updates
    Conflict resolution: [TIMESTAMP]
    Column Mapping: ON, columns INORDER, offset 8, uncomp_len 2064
    Column Name Verifcation: ON
    No Replicated UDT Columns
```

Figure 14-9. onstat -g cat repls Output

The onstat -g con Option

The **onstat -g con** command prints information on conditions and the threads that are waiting for them.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 18:47:42
-- 101376 Kbytes
Conditions with waiters:
cid      addr            name              waiter   waittime
271      c63d930         netnorm           1511     6550
```

Figure 14-10. onstat -g con Output

Output Description:

cid Condition identifier

addr Condition control block address

name Name of condition the thread is waiting on

waiter ID of thread waiting on condition

waittime Time, in seconds, thread has been waiting on this condition

The onstat -g ddr Option

The **onstat -g ddr** command prints the status of the Enterprise Replication database log reader. The **ddr**, or **ddr_snoopy**, is an internal component of Enterprise Replication that reads the log buffers and passes information to the grouper.

You can use the information from the **onstat -g ddr** command to monitor *replay position* in the log file and ensure replay position is never overwritten (which can cause loss of data). The replay position is the point from where, if a system failure occurs, Enterprise Replication starts re-reading the log information into the log update buffers. All the transactions generated before this position at all the target servers have been applied by Enterprise Replication or safely stored in stable queue space.

The **onstat -g ddr** output shows you a snapshot of the replay position, the *snoopy position*, and the *current position*. The snoopy position identifies the position of the **ddr_snoopy** thread in the logical logs. The **ddr_snoopy** has read the log records up until this point. The current position is the position where the server has written its last logical log record.

The *log needs* position is based on replay position and is set at a certain distance from replay position, for example, at seventy percent of the log file. The remainder of the circular log file comprises the DDR BLOCK zone. As messages are acknowledged or stored in the stable queue, the replay position, and hence also the log needs position, should advance. If you notice that replay position is not advancing, this can mean that the stable queue is full or a remote server is down.

If log reading is blocked, data might not be replicated until the problem is resolved. If the block is not resolved, the database server might overwrite the read (**ddr_snoopy**) position, which means that data will not be replicated. If this occurs, you must manually resynchronize the source and target databases.

For servers of Version 9.4, and later, you can enable dynamic log creation by setting the CDR_MAX_DYNAMIC_LOGS configuration parameter in the ONCONFIG file. If the current position reaches the log needs position, instead of going into a blocked state, Enterprise Replication automatically adds another log file. If this option is set, the **onstat -g ddr** command prints the number of dynamic log requests made. For more information, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

Example Output:

The following sample output from the **onstat ddr** command shows the replay position, snoopy position, and current position highlighted.

```
DDR -- Running
# Event  Snoopy   Snoopy   Replay   Replay   Current  Current
Buffers   ID      Position  ID      Position   ID     Position
528       24       165018   24       6a018     24       166000

Log Pages Snooped:      From      From      Tossed
                        Cache      Disk  (LBC full)
                         247      111          0
Total dynamic log requests: 0
DDR events queue

Type   TX id    Partnum  Row id
```

Figure 14-11. onstat -g ddr Output

The onstat -g dic table Option

Without any parameters the **onstat -g dic** option prints one line of information for each table cached in the shared-memory dictionary. If a table name is specified, prints internal SQL information for that table.

For more information see the *Performance Guide*.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 18:47:42
-- 101376 Kbytes

Dictionary Cache:  Number of lists: 31, Maximum list size: 10
list#  size  refcnt  dirty?  heapptr       table name
--------------------------------------------------------
   1     3      1      no     14b5d890   wbe@oninit_shm:informix.t0010url
                1      no     14cbb820   wbe@oninit_shm:informix.t9051themeval
                0      no     14b63c20   wbe@oninit_shm:informix.t0060hits

   2     2      0      no     14b97420   wbe@oninit_shm:informix.t0120import
                1      no     14b6c820   wbe@oninit_shm:informix.t9110domain

   3     3      0      no     14bce020   wbe@oninit_shm:informix.t0150url
                0      no     14d3d820   contact@oninit_shm:informix.wbtags
                0      no     14c87420   wbe@oninit_shm:informix.wbtags

   4     1      0      no     14b7a420   drug@oninit_shm:viagra.product      .....
Total number of dictionary entries: 36
```

Figure 14-12. onstat -g dic Output

Output Description:

list#	Data dictionary hash chain ID
size	Number of entries in this hash
refcnt	Number of SQL statements currently referencing one of the cache entries.
dirty?	Whether the entry has been modified since last written to disk.
heapptr	Address for the heap used to store this table
table name	Name of table in cache

The onstat -g dis Option

Prints a list of database servers and their status, and information about each database server, **INFORMIXDIR**, **sqlhosts** file, ONCONFIG file, and hostname. You can use this option with the database server in any mode, including offline.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 18:47:42
-- 101376 Kbytes

There are 2 servers found
Server        : ol_tuxedo
Server Number : 53
Server Type   : IDS
Server Status : Up
Server Version: IBM Informix Dynamic Server Version 10.00.UC1
Shared Memory : 0xa000000
INFORMIXDIR   : /local1/engines/ol_tuxedo/dist
ONCONFIG      : /local1/engines/ol_tuxedo/dist/etc/onconfig.ol_tuxedo
SQLHOSTS      : /local1/engines/ol_tuxedo/dist/etc/sqlhosts
Host          : avocet

Server        : ol_9next
Server Number : 0
Server Type   : IDS
Server Status : Down
Server Version:
Shared Memory : 0
INFORMIXDIR   : /local1/engines/ol_9next/dist
ONCONFIG      :
SQLHOSTS      :
Host          :
```

Figure 14-13. onstat -g dis Output

Output Description:

Server
: Server name

Server Number
: Number of the server.

Server Type
: Type of server

Server Status
: Up means that the server is online, Down means that the server is offline

Server Version
: Version of the server

Shared Memory
: Location of the shared memory address

INFORMIXDIR
: Location of the the **$INFORMIXDIR/** directory on UNIX and in the **%INFORMIXDIR%** directory on Windows.

ONCONFIG
: Location of the ONCONFIG file

SQLHOSTS
: Location of the **sqlhosts** file

Host
: Host name of the server

The onstat -g dll Option

Prints a list of dynamic libraries that have been loaded.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 18:47:42 -- 101376 Kbytes

Datablades:
addr      slot  vp  baseaddr  filename
140090fc  2     1   fe64d4e0  MYPATH/informix/extend/web.xxxxxx/web.bld
141c70fc        2   fe7cd4e0
141ca0fc        3   fe7cd4e0
```

Figure 14-14. onstat -g dll Output

Output Description:

addr	DLL address
slot	Slot number entry in the library table
vp	Virtual processor ID
baseaddr	Virtual processor base address
filename	DLL filename

The onstat -g dri Option

The **onstat -g dri** option prints information about High-Availability Data Replication on the current server.

Example Output:

```
Data Replication:
  Type         State        Paired server        Last DR CKPT (id/pg)
  primary      off          amit_secondary               -1 / -1

  DRINTERVAL   2
  DRTIMEOUT    30
  DRAUTO       0
  DRLOSTFOUND  /vobs/tristarm/sqldist/etc/dr.lostfound
  DRIDXAUTO    0
.
```

Figure 14-15. onstat -g dri Output

Output Description:

Type	Current type of server: primary, secondary, or standard
State	on or off
Paired server	Name of the primary or secondary server that this server is paired with
Last DR CKPT	Last checkpoint ID and page

The second section lists the values of the following configuration parameters in the ONCONFIG file:

- DRINTERVAL
- DRTIMEOUT
- DRAUTO
- DRLOSTFOUND
- DRIDXAUTO

The onstat -g dsc Option

Prints a list of distribution cache information.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:54:52
-- 101376 Kbytes
Distribution Cache:
    Number of lists             : 31
    PC_POOLSIZE                 : 50
    Number of entries           : 0
    Number of entries in use    : 0
Distribution Cache Entries:
list#  id  ref_cnt   dropped?   heap_ptr        distribution name
-----------------------------------------------------------------
Distribution Cache is empty.
```

Figure 14-16. onstat -g dsc Output

Output Description:

The first section of output describes the distribution cache.

Number of lists	Number of lists in the distribution cache
PC_POOLSIZE	Number of entries that can be cached at one time
Number of entries	Number of entries in the distribution cache
Number of entries in use	Number of entries being used

The second section of output describes the distribution cache entries.

list#	Distribution cache hash chain ID
id	Number of hash entries
ref_cnt	Number of statements referencing a cache entry
dropped?	Whether this entry has been dropped since being added to the cache
heap_ptr	Heap address used to store this entry
distribution name	The name of the distribution in the cache

The onstat -g dss Option

The **onstat -g dss** command prints detailed statistical information about the activity of individual data sync threads in an Enterprise Replication environment. The data sync thread applies the transaction on the target server. Statistics include the number of applied transactions and failures and when the last transaction from a source was applied.

The **onstat -g dss** command has the following formats:

```
onstat -g dss
onstat -g dss modifier
```

The following table describes the values for *modifier*.

Modifier	Action
UDR	Prints summary information about any UDR invocations by the data sync threads.
UDRx	Prints expanded information (including a summary of error information) about any UDR invocations by the data sync threads. The `Procid` column lists the UDR procedure ID.

Example Output:

In the following example, only one data sync thread is currently processing the replicated data. It has applied a total of one replicated transaction and the transaction was applied at 2004/09/13 18:13:10. The Processed Time field shows the time when the last transaction was processed by this data sync thread.

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 00:00:28 -- 28672 Kbytes
DS thread statistic
cmtTime      Tx       Tx        Tx      Last Tx
Name         < local  Committed Aborted Processed  Processed Time
------------ -------  --------- ------- ---------  -----------------
CDRD_1             0         1       0         1  (1095117190) 2004/09/13 18:13:10
            Tables (0.0%):
            Databases: test
CDR_DSLOCKWAIT = 1
CDR_DSCLOSEINTERVAL = 60
```

Figure 14-17. onstat -g dss Output

The onstat -g dtc Option

The **onstat -g dtc** command prints statistics about the delete table cleaner. The delete table cleaner removes rows from the delete table when they are no longer needed.

The **-g dtc** option is used primarily as a debugging tool and by Technical Support.

Example Output:

In the following example, the thread name of the delete table cleaner is **CDRDTCleaner**. The total number of rows deleted is **1**. The last activity on this thread occurred at 2004/09/13 18:47:19. The delete table for replicate **rep1** was last cleaned at 2004/09/13 18:28:25.

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 00:59:15 -- 28672 Kbytes
-- Delete Table Cleanup Status as of (1095119368) 2004/09/13 18:49:28
      thread           = 49 <CDRDTCleaner>
      rows deleted     = 1
      lock timeouts    = 0
      cleanup interval = 300
      list size        = 3
      last activity    = (1095119239) 2004/09/13 18:47:19
Id      Database                         Last Cleanup Time
        Replicate           Server             Last Log Change
==========================================================
000001  test                             (1095118105) 2004/09/13 18:28:25
        rep1                g_bombay           (1095118105) 2004 /09/1318:28:25
        rep1                g_delhi            (1095118105) 2004 /09/13 18:28:25
000002  test                             <never cleaned>
```

Figure 14-18. onstat -g dtc Output

The onstat -g env Option

The **onstat -g env** option displays the values of environment variables the database server currently uses. You can specify one of the following invocations.

Invocation	Explanation
onstat -g env	Displays the settings of variables when the database server was started Does not display variables that have not been set explicitly.
onstat -g env *sessionid*	Displays the settings that a specific session uses. This display includes the following values: • Set in the environment of the session • Assigned by the database server, as **onstat -g env** displays
onstat -g env all	Displays the settings used by all sessions This display is the same as the output of **onstat -g env** and **onstat -g env** *sessionid* iteratively on all current sessions.
onstat -g env *variable*	Displays the default value of the specified variable This ***variable*** argument eliminates the need to pipe the output to **grep** (or some other utility) to locate a variable among many that might be set.
onstat -g env *sessionid variable*	Displays the value of the specified variable that the specified session uses The ***sessionid*** and ***variable*** arguments eliminate the need to pipe the output to **grep** (or some other utility) to locate a variable among many that might be set.

You might want to display the values of environment variables in any of the following situations:

- The database server instance has been up for months, and you cannot remember the setting of an environment variable (such as the server locale setting **SERVER_LOCALE**).

- You want to display the complete list of values for a variable to identify when a variable has been set in multiple places.
- Environment files on disk might have changed or been lost in the interim.
- A support engineer wants to know settings of specific environment variables.

The **onstat -g env** option displays the current setting of a variable and the complete list of values each time the variable was set in the environment. For example, if PDQPRIORITY is set to 10 in the **.informix.rc** file and set to 55 in the shell environment, **onstat -g env** displays both values.

However, if you change the PDQPRIORITY with the **onmode -q pdqpriority** *sessionid* option, **onstat -g env** does not display the new value for the session. The **onstat -g env** option displays only the values of variables set in the environment. It does not display values modified while the session is running.

The following figure show the output for the **onstat -g env** option

```
# onstat -g env

IBM Informix Dynamic Server Version 9.40.UC1 -- On-Line -- Up 4 days 17:08:43
-- 45056 Kbytes

Variable            Value [values-list]
DBDATE              DMY4/
DBDELIMITER         |
DBPATH              .
DBPRINT             lp -s
DBTEMP              /tmp
INFORMIXDIR         /build2/9.30/tristarm/sqldist
                    [/build2/9.30/tristarm/sqldist]
                    [/usr/informix]
INFORMIXSERVER      parata930
INFORMIXTERM        termcap
LANG                C
LC_COLLATE          C
LC_CTYPE            C
LC_MONETARY         C
LC_NUMERIC          C
LC_TIME             C
LD_LIBRARY_PATH     /usr/openwin/lib:/lib:/usr/lib
LKNOTIFY            yes
LOCKDOWN            no
NODEFDAC            no
NON_M6_ATTRS_OK     1
PATH                         /build2/9.30/tristarm/sqldist/bin:.:
                        /root/bin:/opt/SUNWspro/bin:/usr/ccs/bin:
                        /usr/openwin/bin:/usr/sbin:/usr/bin:/usr
                     /local/binSERVER_LOCALE       en_US.819
SHELL               /bin/ksh
SINGLELEVEL         no
SUBQCACHESZ         10
TBCONFIG            onconfig
TERM                xterm
                    [xterm]
                    [dumb]
TERMCAP             /etc/termcap
TZ                  GB
```

Figure 14-19. onstat -g env Output

The onstat -g ffr pool name | session id Option

The **onstat -g ffr** option prints free fragments for a pool of shared memory.

Example Output:

```
Free list for pool name dfm_pool:
addr            size
10ac92f20       224
10ac8c0c0       3904
```

Figure 14-20. onstat -g ffr Output

Output Description:

addr Pool fragment address

size Fragment size, in bytes

The onstat -g glo Option

The **onstat -g glo** option prints global multithreading information.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:55:02 -- 101376 Kbytes
MT global info:
sessions threads  vps      lngspins
0        49       14       1
          sched calls     thread switches yield 0   yield n   yield forever
total:    900100          898846          1238      27763     423778
per sec:  327             325             2         12        151
Virtual processor summary:
 class       vps       usercpu   syscpu    total
 cpu         4         0.92      0.10      1.02
 aio         4         0.02      0.02      0.04
 lio         1         0.00      0.00      0.00
 pio         1         0.00      0.00      0.00
 adm         1         0.00      0.01      0.01
 msc         1         0.00      0.00      0.00
 fifo        2         0.00      0.00      0.00
 total       14        0.94      0.13      1.07
Individual virtual processors:
 vp    pid       class       usercpu   syscpu    total
 1     2599      cpu         0.25      0.06      0.31
 2     2602      adm         0.00      0.01      0.01
 3     2603      cpu         0.23      0.00      0.23
 4     2604      cpu         0.21      0.03      0.24
 5     2605      cpu         0.23      0.01      0.24
 6     2606      lio         0.00      0.00      0.00
 7     2607      pio         0.00      0.00      0.00
 8     2608      aio         0.02      0.02      0.04
 9     2609      msc         0.00      0.00      0.00
 10    2610      fifo        0.00      0.00      0.00
 11    2611      fifo        0.00      0.00      0.00
 12    2612      aio         0.00      0.00      0.00
 13    2613      aio         0.00      0.00      0.00
 14    2614      aio         0.00      0.00      0.00
                 tot         0.94      0.13      1.07
```

Figure 14-21. onstat -g glo Output

Output Description:

Virtual Processor Summary

class	The type of virtual processor
vps	The number of instances of this class of VP
usercpu	The total user time this class of VP has spent running on the CPU in seconds
syscpu	The total system time this class of VP has spent running on the CPU in seconds
total	The total number of virtual processors, user time and system time

Individual Virtual Processors

vp	Virtual processor number
pid	Process ID of this oninit process
class	Virtual processor class
usercpu	Total user time the VP has run on the CPU
syscpu	Total system time the VP has run on the CPU
total	Total number of VPs, user time, and system time

The onstat -g grp Option

The **onstat -g grp** command prints statistics about the Enterprise Replication grouper. The grouper evaluates the log records, rebuilds the individual log records into the original transaction, packages the transaction, and queues the transaction for transmission.

The **-g grp** option is used primarily as a debugging tool and by Technical Support.

The **onstat -g grp** command has the following formats:

```
onstat -g grp
onstat -g grp modifier
```

The following table describes the values for *modifier*.

Modifier	Action
A	Prints all the information printed by the G, T, P, E, R, and S modifiers
E	Prints grouper evaluator statistics
Ex	Prints grouper evaluator statistics, expands user-defined routine (UDR) environments
G	Prints grouper general statistics
L	Prints grouper global list
Lx	Prints grouper global list, expands open transactions
M	Prints grouper compression statistics
Mz	Clears grouper compression statistics
P	Prints grouper table partition statistics
pager	Prints grouper paging statistics
R	Prints grouper replicate statistics
S	Prints grouper serial list head (The serial list head is the first transaction in the list, that is, the next transaction that will be placed in the send queue.)
Sl	Prints grouper serial list (The serial list is the list of transactions, in chronological order.)
Sx	Prints grouper serial list, expands open transactions
T	Prints grouper transaction statistics
UDR	Prints summary information about any UDR invocations by the grouper threads
UDRx	Prints expanded information (including a summary of error information) about any UDR invocations by the grouper threads The `Procid` column lists the UDR procedure ID.

Example Output:

This section contains sample output from various **onstat -g grp** *modifier* commands. The following sample shows output for the **onstat -g grp** command.

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 01:47:07 -- 28672 Kbytes
Grouper at 0xb014018:
Last Idle Time: (1095122236) 2004/09/13 19:37:16
RSAM interface ring buffer size: 528
RSAM interface ring buffer pending entries: 0
Eval thread interface ring buffer size: 48
Eval thread interface ring buffer pending entries: 0
Log update buffers in use: 0
Max log update buffers used at once: 5
Log update buffer memory in use: 0
Max log update buffer memory used at once: 320
Updates from Log: 16
Log update links allocated: 512
Blob links allocated: 0
Conflict Resolution Blocks Allocated: 0
Memory pool cache: Empty
Last Tx to Queuer began : (1095118105) 2004/09/13 18:28:25
Last Tx to Queuer ended : (1095118105) 2004/09/13 18:28:25
Last Tx to Queuer log ID, position: 12,23
Open   Tx: 0
Serial Tx: 0
Tx not sent: 0
Tx sent to Queuer: 2
Tx returned from Queuer: 2
Events sent to Queuer: 7
Events returned from Queuer: 7
Total rows sent to Queuer: 2
Open Tx array size: 1024
Table 'tab' at 0xae8ebb0 [ CDRShadow ]
Table 'tab12' at 0xae445e0 [ CDRShadow ]
```

Figure 14-22. onstat -g grp Output (Part 1 of 3)

```
Grouper Table Partitions:
  Slot  312...
    'tab' 1048888
  Slot  770...
    'tab12' 3145730
  Slot 1026...
    'tab12' 4194306
Repl links on global free list: 2
Evaluators: 3
  Evaluator at 0xb03d030 ID 0 [Idle:Idle] Protection:unused
    Eval iteration: 1264
    Updates evaluated: 0
    Repl links on local free list: 256
    UDR environment table at 0xb03d080
        Number of environments:         0
        Table memory limit    :      25165
        Table memory used     :         0
        SAPI memory limit     :     131072
        SAPI memory used      :         0
        Count failed UDR calls:         0
  Evaluator at 0xb03d0d8 ID 1 [Idle:Idle] Protection:unused
    Eval iteration: 1265
    Updates evaluated: 2
    Repl links on local free list: 254
    UDR environment table at 0xb03d128
        Number of environments:         0
        Table memory limit    :      25165
        Table memory used     :         0
        SAPI memory limit     :     131072
        SAPI memory used      :         0
        Count failed UDR calls:         0
  Evaluator at 0xb03d180 ID 2 [Idle:Idle] Protection:unused
    Eval iteration: 1266
    Updates evaluated: 4
    Repl links on local free list: 256
    UDR environment table at 0xb03d1d0
        Number of environments:         0
        Table memory limit    :      25165
        Table memory used     :         0
        SAPI memory limit     :     131072
        SAPI memory used      :         0
        Count failed UDR calls:         0
        Total Free Repl links 768
```

Figure 14-22. onstat -g grp Output (Part 2 of 3)

```
Replication Group 6553601 at 0xb0a8360
  Replication at 0xb0a82b0 6553601:6553601 (tab) [ NotifyDS FullRowOn ]
    Column Information [ CDRShadow VarUDTs InOrder Same ]
      CDR Shadow: offset 0, size 8
      In Order: offset 8, size 10
Replication Group 6553602 at 0xb0a8480
  Replication at 0xb0a83d0 6553602:6553602 (tab12)  [ Ignore Stopped NotifyDS FullRowOn ]
    Column Information [ CDRShadow VarUDTs InOrder Same ]
      CDR Shadow: offset 0, size 8
      In Order: offset 8, size 16
```

Figure 14-22. onstat -g grp Output (Part 3 of 3)

The following example shows output for the **onstat -g grp E** command. The field **Evaluators: 4** indicates that there are four evaluation threads configured for the system.

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 02:07:10 -- 36864 Kbytes
Repl links on global free list: 0 Evaluators: 4
  Evaluator at 0xba71840 ID 0 [Idle:Idle] Protection: unused
    Eval iteration: 1007
    Updates evaluated: 0
    Repl links on local free list: 256
    UDR environment table at 0xba71890
        Number of environments:          0
        Table memory limit    :      16777
        Table memory used     :          0
        SAPI memory limit     :     131072
        SAPI memory used      :          0
        Count failed UDR calls:          0
  Evaluator at 0xba718f0 ID 1 [Idle:Idle] Protection: unused
    Eval iteration: 1007
    Updates evaluated: 0
    Repl links on local free list: 256
    UDR environment table at 0xba71940
        Number of environments:          0
        Table memory limit    :      16777
        Table memory used     :          0
        SAPI memory limit     :     131072
        SAPI memory used      :          0
        Count failed UDR calls:          0
```

Figure 14-23. onstat -g grp E Output (Part 1 of 2)

```
  Evaluator at 0xba8c260 ID 2 [Idle:Idle] Protection: unused
    Eval iteration: 1007
    Updates evaluated: 0
    Repl links on local free list: 256
    UDR environment table at 0xba8c2b0
        Number of environments:         0
        Table memory limit    :     16777
        Table memory used     :         0
        SAPI memory limit     :    131072
        SAPI memory used      :         0
        Count failed UDR calls:         0
  Evaluator at 0xbaac2a0 ID 3 [Idle:Idle] Protection: unused
    Eval iteration: 1007
    Updates evaluated: 0
    Repl links on local free list: 256
    UDR environment table at 0xbaac2f0
        Number of environments:         0
        Table memory limit    :     16777
        Table memory used     :         0
        SAPI memory limit     :    131072
        SAPI memory used      :         0
        Count failed UDR calls:         0
Total Free Repl links 1024
```

Figure 14-23. onstat -g grp E Output (Part 2 of 2)

The following example shows output for the **onstat -g grp G** command.

```
IBM Informix Dynamic Server Version 10.00.UC1     -- On-Line -- Up 02:08:56 -- 36864 Kbytes
Grouper at 0xb8ab020:
Last Idle Time: (1095115397) 2004/09/13 17:43:17
RSAM interface ring buffer size: 1040
RSAM interface ring buffer pending entries: 0
Eval thread interface ring buffer size: 64
Eval thread interface ring buffer pending entries: 0
Log update buffers in use: 0
Max log update buffers used at once: 1
Log update buffer memory in use: 0
Max log update buffer memory used at once: 64
Updates from Log: 1
Log update links allocated: 512
Blob links allocated: 0
Conflict Resolution Blocks Allocated: 0
Memory pool cache: Empty
```

Figure 14-24. onstat -g grp G Output

The following example shows output for the **onstat -g grp P** command. In the example, the grouper is evaluating rows for the **account**, **teller** and **customer** tables.

```
IBM Informix Dynamic Server Version 10.00.UC1     -- On-Line -- Up 02:11:39 -- 36864 Kbytes
Table 'teller' at 0xb851480 [ CDRShadow VarChars ]
Table 'account' at 0xb7faad8 [CDRShadow VarChars VarUDTs Floats Blobs]
Table 'customer' at 0xbbe67a8 [CDRShadow VarChars VarUDTs]
Grouper Table Partitions:
  Slot  387...
    'account' 1048707
  Slot  389...
    'teller' 1048709
  Slot  394...
    'customer' 1048714
```

Figure 14-25. onstat -g grp P Output

The following example shows output for the **onstat -g grp pager** command. The sample output shows the grouper large transaction evaluation statistics.

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 00:20:42 -- 28672 Kbytes
Grouper Pager statistics:
Number of active big transactions: 0
Total number of big transactions processed: 0
Spool size of the biggest transaction processed: 0 Bytes
```

Figure 14-26. onstat -g grp pager Output

The following example shows output for the **onstat -g grp R** command. In this example, the grouper is configured to evaluate rows for replicates with IDs **6553601** and **6553602** (you can use the **onstat -g cat repls** command to obtain the replicate names). The **Ignore** attribute of replicate ID **6553602** shows that the grouper is currently not evaluating rows for this replicate. This can happen if the replicate state is not ACTIVE. You can obtain the replicate state using the **onstat -g cat repls** command.

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 00:04:47 -- 28672 Kbytes
Replication Group 6553601 at 0xb0a8360
  Replication at 0xb0a82b0 6553601:6553601 (tab) [ NotifyDS FullRowOn ]
    Column Information [ CDRShadow VarUDTs InOrder Same ]
      CDR Shadow: offset 0, size 8
      In Order: offset 8, size 10
Replication Group 6553602 at 0xb0a8480
  Replication at 0xb0a83d0 6553602:6553602 (tab12) [ Ignore Stopped NotifyDS FullRowOn ]
    Column Information [ CDRShadow VarUDTs InOrder Same ]
      CDR Shadow: offset 0, size 8
      In Order: offset 8, size 16
```

Figure 14-27. onstat -g grp R Output

The following example shows output for the **onstat -g grp T** command. In this example, the grouper evaluated and queued 1 transaction to the send queue. The **Tx sent to Queuer** field shows the total number of transactions evaluated and queued to the send queue for propagating to all the replicate participants. The **Total rows sent to Queuer** field shows the total number of rows queued to the send queue for propagating to all the replicate participants.

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 00:14:51 -- 28672 Kbytes
Last Tx to Queuer began : (1095116676) 2004/09/13 18:04:36
Last Tx to Queuer ended : (1095116676) 2004/09/13 18:04:36
Last Tx to Queuer log ID, position: 5,3236032
Open   Tx: 0
Serial Tx: 0
Tx not sent: 0
Tx sent to Queuer: 1
Tx returned from Queuer: 0
Events sent to Queuer: 0
Events returned from Queuer: 0
Total rows sent to Queuer: 1
Open Tx array size: 1024
```

Figure 14-28. onstat -g grp T Output

The onstat -g ioa Option

The **onstat -g ioa** option prints combined information from **-g ioq**, **-g iov**, and **-g iob**.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:55:13 -- 101376 Kbytes
AIO global info:
   7 aio classes
   4 open files
  64 max global files
32768 max files from setrlimit
AIO I/O queues:
q name/id    len maxlen totalops  dskread dskwrite  dskcopy
 fifo   0      0      0        0        0        0        0
  adt   0      0      0        0        0        0        0
  msc   0      0      0        0        0        0        0
  aio   0      0      0        0        0        0        0
  pio   0      0      0        0        0        0        0
  lio   0      0      0        0        0        0        0
  gfd   3      0      1      607        0      607        0
AIO I/O vps:
class/vp s io/s  totalops dskread dskwrite dskcopy wakeups io/wup polltries
pollfound kaio_pend
fifo  0  i 0.0   0        0       0        0       0        0.0   0
0        0
fifo  1  i 0.0   0        0       0        0       0        0.0   0
0        0
msc   0  i 0.0   0        0       0        0       0        0.0   0
0        0
aio   0  i 0.3   607      0       607      0       607      1.0   0
0        0
AIO global files:
gfd pathname         totalops  dskread dskwrite  io/s
  3 rootdbs.1            607        0      607   0.3
AIO big buffer usage summary:
class                reads                                   writes
       pages    ops  pgs/op  holes  hl-ops hls/op      pages    ops  pgs/op
fifo      0       0   0.00      0       0   0.00          0      0   0.00
 kio      0       0   0.00      0       0   0.00          0      0   0.00
 adt      0       0   0.00      0       0   0.00          0      0   0.00
 msc      0       0   0.00      0       0   0.00          0      0   0.00
 aio      0       0   0.00      0       0   0.00        607    607   1.00
 pio      0       0   0.00      0       0   0.00          0      0   0.00
 lio      0       0   0.00      0       0   0.00          0      0   0.00
```

Figure 14-29. onstat -g ioa Output

Output Description:

For a description of each output column, see the **-g ioq**, **-g iov**, and **-g iob** options.

The onstat -g iob Option

The **onstat -g iob** option prints a summary of big buffer use.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:55:13 -- 101376 Kbytes

AIO big buffer usage summary:
class                reads                                   writes
       pages    ops  pgs/op  holes  hl-ops hls/op      pages    ops  pgs/op
fifo      0       0   0.00      0       0   0.00          0      0   0.00
 kio      0       0   0.00      0       0   0.00          0      0   0.00
 adt      0       0   0.00      0       0   0.00          0      0   0.00
 msc      0       0   0.00      0       0   0.00          0      0   0.00
 aio      0       0   0.00      0       0   0.00        607    607   1.00
 pio      0       0   0.00      0       0   0.00          0      0   0.00
 lio      0       0   0.00      0       0   0.00          0      0   0.00
```

Figure 14-30. onstat -g iob Output

The onstat -g iof Option

The **onstat -g iof** option prints asynchronous I/O statistics by chunk or file. This option is similar to the **-D** option, except that information on nonchunk files is also displayed. It includes information about temporary files and sort-work files.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:55:32 -- 101376 Kbytes
AIO global files:
gfd pathname         totalops  dskread dskwrite  io/s
  3 rootdbs.1             613        0      613   0.3
```

Figure 14-31. onstat -g iof Output

Output Description:

gfd	Global file descriptor number for this chunk
pathname	The pathname of the chunk
totalops	Total number of read and write operations that have occurred against the chunk
dskread	Number of disk read that have occurred against the chunk
dskwrite	Number of disk writes that have occurred against the chunk
io/s	Number of I/Os per second

The onstat -g iog Option

The **onstat -g iog** option prints AIO global information.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:55:42 -- 101376 Kbytes
AIO global info:
   7 aio classes
   4 open files
  64 max global files
32768 max files from setrlimit
```

Figure 14-32. onstat -g iog Output

The onstat -g ioq queue name Option

The **onstat -g ioq** option shows statistics about the number and types of operations performed by I/O queues. If a *queue_name* is given then only queues with that name are shown. If no *queue_name* is given then information is given for all queues.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 01:00:54 -- 109568 Kbytes

AIO I/O queues:
q name/id    len maxlen totalops  dskread dskwrite  dskcopy
sqli_dbg   0      0      0        0        0        0        0
 fifo   0      0      0        0        0        0        0
  adt   0      0      0        0        0        0        0
  msc   0      0      1      537        0        0        0
  aio   0      0      3     6537      238     5777        0
  pio   0      0      2     1103        0     1102        0
  lio   0      0      2    11795        0    11794        0
  gfd   3      0     17    17489     1526    15963        0
  gfd   4      0     17    18347     2384    15963        0
  gfd   5      0     16      220       41      179        0
  gfd   6      0      4        4        0        4        0
  gfd   7      0      4        4        0        4        0
  gfd   8      0      4        4        0        4        0
  gfd   9      0      9       54       24       30        0
  gfd  10      0     16      149       40      109        0
  gfd  11      0     16      621      128      493        0
  gfd  12      0     16     1953     1146      807        0
  gfd  13      0     16      409       71      338        0
  gfd  14      0     16      378       60      318        0
```

Figure 14-33. onstat -g ioq Output

Output Description:

q name/id
: The name and number of the I/O queue. The name indicates what type of queue it is. The number is used to tell queues of the same name apart.

  Here is a list of the possible queue names and what each type of queue handles:

  sqli_dbg — Handles I/O for IBM Technical Support's SQL Interface Debugging feature

  fifo — Handles I/O for FIFO VPs

  adt — Handles auditing I/O

msc Handles miscellaneous I/O

aio Handles IBM Informix asynchronous I/O

kio Handles kernel AIO

pio Handles physical logging I/O

lio Handles logical logging I/O

gfd Global File Descriptor - Each primary and mirror chunk is given a separate global file descriptor. Individual gfd queues are used depending on whether kaio is on and the associated chunk is cooked or raw.

len The number of pending I/O requests in the queue

maxlen The largest number of I/O requests that have been in the queue at the same time

totalops The total number of I/O operations that have been completed for the queue

dskread Total number of completed read operations for the queue

dskwrite Total number of completed write operations for the queue

dskcopy Total number of completed copy operations for the queue

The onstat -g iov Option

The **onstat -g iov** option shows asynchronous I/O statistics for each virtual processor.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:56:26 -- 101376 Kbytes
AIO I/O vps:
class/vp s  io/s totalops dskread dskwrite dskcopy  wakeups io/wup polltries pollfound kaio_pend
fifo  0  i  0.0  0        0       0        0        0       0.0    0         0         0
fifo  1  i  0.0  0        0       0        0        0       0.0    0         0         0
msc   0  i  0.0  0        0       0        0        0       0.0    0         0         0
aio   0  s  0.3  628      0       628      0        628     1.0    0         0         0
```

Figure 14-34. onstat -g iov Output

Output Description:

class The class of the virtual processor.

vp The ID number of the virtual processor within its class.

s Current status of the AIO virtual processor

f Fork

i Idle

s Search

b Busy

o Open

c Close

io/s The average I/O speed (measured in operations per second) for the virtual processor since the time the database server started or since **onstat -z** was last run, whichever happened last.

totalops Total number of I/O operations performed by this virtual processor since the time the database server started or since **onstat -z** was last run, whichever happened last.

dskread Total number of read operations performed by this virtual processor since the time the database server started or since **onstat -z** was last run, whichever happened last.

dskwrite Total number of write operations performed by this virtual processor since the time the database server started or since **onstat -z** was last run, whichever happened last.

dskcopy Total number of copy operations performed by this virtual processor since the time the database server started or since **onstat -z** was last run, whichever happened last.

wakeups For AIO VPs, the number of times the virtual processor has gone idle since the time the database server started or since **onstat -z** was last run, whichever happened last.

io/wup For AIO VPs, the average number of I/O operations performed per wake-up by this virtual processor since the time the database server started or since **onstat -z** was last run, whichever happened last.

polltries For AIO VPs, the total number of times the kaio thread running on this virtual processor checked the operating system to see if I/O it had requested was done. Count is from the time the database server started or since **onstat -z** was last run, whichever happened last.

pollfound For AIO VPs, the total number of times the kaio thread running on this virtual processor checked the operating system to see if I/O it had requested was done and found that the I/O was complete. Count is from the time the database server started or since **onstat -z** was last run, whichever happened last.

kaio_pend For AIO VPs, the number of I/O requests made by the kaio thread that have not yet completed.

The onstat -g lmx Option

The **onstat -g lmx** option prints all locked mutexes.

Example Output:

```
Locked mutexes:
mid      addr             name              holder   lkcnt  waiter   waittime
Number of mutexes on VP free lists: 49
```

Figure 14-35. onstat -g lmx Output

Output Description:

mid Internal mutex identifier

addr Address of locked mutex

name	Name of the mutex
holder	Session ID of the thread holding the mutex
lkcnt	Number of waiters for this mutex
waiter	List of addresses waiting for this mutex
waittime	Amount of time this thread has been waiting

The onstat -g mem pool name | session id Option

The **onstat -g mem** option prints memory statistics for a pool. Session pools are named with the session number. If no argument is provided, information about all pools is displayed.

Example Output:

```
Pool Summary:
name         class addr            totalsize freesize #allocfrag #freefrag
resident     R     10a001028       2420736   7960     2          2
res-buff     R     10a250028       8269824   7960     2          2
global       V     10aac0028       9351168   32648    650        11
...
...
...
onmode_mon   V     10b983028       20480     2752     108        1
13           V     10bd5d028       16384     5200     12         2
Blkpool Summary:
name         class addr            size      #blks      pre-hint   szavail
global       V     10aac8920       0         0          0          0
xmf_msc_pl   V     10ac84ca0       954368    73         0          0
```

Figure 14-36. onstat -g mem Output

Output Description:

Pool Summary

name	Pool name
class	Shared memory segment type where pool is created
addr	Pool memory address
totalsize	Pool size, in bytes
freesize	Free memory in pool
#allocfrag	Allocated fragments in pool
#freefrag	Free fragments in pool

Blkpool Summary

name	Pool name
class	Shared memory segment type where pool is created
addr	Pool memory address
size	Pool size, in bytes
#blks	Number of blocks in pool

The onstat -g mgm Option

The **onstat -g mgm** option prints Memory Grant Manager (MGM) resource information. You can use the **onstat -g mgm** option to monitor how MGM coordinates memory use and scan threads. This **onstat** option reads shared-memory structures and provides statistics that are accurate at the instant that the command executes.

The **onstat -g mgm** output displays a unit of memory called a *quantum.* The *memory quantum* represents a unit of memory, as follows:

```
memory quantum = DS_TOTAL_MEMORY / DS_MAX_QUERIES
```

The following calculation shows the memory quantum for the values that Figure 14-37 displays:

```
memory quantum = 4000 kilobytes / 31
               = 129 kilobytes
```

The *scan thread quantum* is always equal to 1.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1  -- On-Line -- Up 00:00:51 -- 21504 Kbytes

Memory Grant Manager (MGM)
--------------------------
MAX_PDQPRIORITY:  100
DS_MAX_QUERIES:    31
DS_MAX_SCANS:      1048576
DS_NONPDQ_QUERY_MEM: 128 KB
DS_TOTAL_MEMORY:   4000 KB

Queries:   Active     Ready   Maximum
                        0        0       31
Memory:     Total      Free   Quantum
(KB)         4000      4000       128

Scans:      Total      Free   Quantum
               1048576   1048576        1

Load Control:    (Memory)      (Scans)  (Priority)  (Max Queries)   (Reinit)
                       Gate 1       Gate 2      Gate 3         Gate 4     Gate 5
(Queue Length)          0           0           0              0          0

Active Queries:  None
Ready Queries:  None
Free Resource       Average #       Minimum #
--------------    ---------------     ---------
Memory              0.0 +- 0.0          500
Scans               0.0 +- 0.0         1048576

Queries             Average #       Maximum #    Total #
--------------    ---------------     ---------    -------
Active              0.0 +- 0.0            0         0
Ready               0.0 +- 0.0            0         0

Resource/Lock Cycle Prevention count:  0
```

Figure 14-37. onstat -g mgm Output

Output Description:

The first portion of the output shows the values of the PDQ configuration parameters.

The second portion of the output describes MGM internal control information. It includes four groups of information. The first group is **Queries**:

Active
: Number of PDQ queries that are currently executing

Ready
: Number of user queries ready to run but whose execution the database server deferred for load-control reason

Maximum
: Maximum number of queries that the database server permits to be active. Reflects current value of the DS_MAX_QUERIES configuration parameter

The next group is **Memory**:

Total
: Kilobytes of memory available for use by PDQ queries (DS_TOTAL_MEMORY specifies this value.)

Free
: Kilobytes of memory for PDQ queries not currently in use

Quantum
: Kilobytes of memory in a memory quantum

The next group is **Scans**:

Total
: The total number of scan threads as specified by the DS_MAX_SCANS configuration parameter

Free
: Number of scan threads currently available for decision-support queries

Quantum
: The number of scan threads in a scan-thread quantum

The last group in this portion of the output describes MGM **Load Control**:

Memory
: Number of queries that are waiting for memory

Scans
: Number of queries that are waiting for scans

Priority
: Number of queries that are waiting for queries with higher PDQ priority to run

Max Queries
: Number of queries that are waiting for a query slot

Reinit
: Number of queries that are waiting for running queries to complete after an **onmode -M** or **-Q** command

The next portion of the output, **Active Queries**, describes the MGM active and ready queues. This portion of the output shows the number of queries waiting at each gate:

Session
: The session ID for the session that initiated the query

Query
: Address of the internal control block associated with the query

Priority
: PDQ priority assigned to the query

Thread	Thread that registered the query with MGM
Memory	Memory currently granted to the query or memory reserved for the query (Unit is MGM pages, which is 8 kilobytes.)
Scans	Number of scan threads currently used by the query or number of scan threads allocated to the query
Gate	Gate number at which query is waiting

The next portion of the output, **Free Resource**, provides statistics for MGM free resources. The numbers in this portion and in the final portion reflect statistics since system initialization or the last **onmode -Q**, **-M**, or **-S** command. This portion of the output contains the following information:

Average	Average amount of memory and number of scans
Minimum	Minimum available memory and number of scans

The last portion of the output, **Queries**, provides statistics concerning MGM queries:

Average	Average active and ready queue length
Maximum	Maximum active and ready queue length
Total	Total active and ready queue length

The onstat -g nbm Option

The **onstat -g nbm** option shows the block bit map for the nonresident segments. Each bit of the bitmap represents a 4 KB block. If the block is used then the bit is set to 1. If it is free the bit is set to 0. The bitmap is shown as a series of hexadecimal numbers. The bits, and therefore the blocks, are numbered starting at 0 so the first block is block 0, the second is block 1, and so on.

Example Output:

This example shows the bitmap for the segment of virtual memory at 0x10CC00000. The bitmap itself is at 0x10CC00290. All 1792 blocks of the segment are free except for block 0 and block 1023.

```
Block bitmap for virtual segment address 0x10cc00000:
 address = 0x10cc00290, size(bits) = 1792
 used = 1, largest_free = -1
      0:8000000000000000 0000000000000000 0000000000000000 0000000000000000
    256:0000000000000000 0000000000000000 0000000000000000 0000000000000000
    512:0000000000000000 0000000000000000 0000000000000000 0000000000000000
    768:0000000000000000 0000000000000000 0000000000000000 0000000000000001
   1024:0000000000000000 0000000000000000 0000000000000000 0000000000000000
   1280:0000000000000000 0000000000000000 0000000000000000 0000000000000000
   1536:0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

Figure 14-38. onstat -g nbm Output

Output Description:

address	The starting address of the bitmap.

size The number of bits in the bitmap. This is also the number of 4 KB blocks in the memory segment.

used The total number of bits in the bitmap that are set to 1. This is also the number of 4 KB blocks that are in use in the memory segment.

largest free If this is a value other than -1 it is the largest number of consecutive bits that are free, which is also the number of 4 KB blocks in the largest contiguous set of blocks in the memory segment.

A value of -1 means that the largest free space has not been calculated. The database server only calculates the largest free space if it tries to allocate a set of blocks starting at the *lastalloc* block but there is not enough free space. The value is set to -1 again as soon as another block is allocated in the segment.

The onstat -g nif Option

The **onstat -g nif** command prints statistics about the network interface for Enterprise Replication. The output shows which sites are connected and provides a summary of the number of bytes sent and received by each site. This can help you determine that a site is in a hung state, if it is not sending or receiving bytes.

The **-g nif** option is used primarily as a debugging tool and by Technical Support.

The **onstat -g nif** command has the following formats:

```
onstat -g nif
onstat -g nif modifier
```

The following table describes the values for *modifier*.

Modifier	Action
all	Prints the sum and the sites
sites	Prints the NIF site context blocks
serverid	Prints information about the replication server whose groupID is serverID
sum	Prints the sum of the number of buffers sent and received for each site

Example Output:

The following example shows output for the **onstat -g nif** command. In this example, the local server is connected to the server group **g_bombay** and its CDR ID is **200**. The connection status is set to running. The server group **g_bombay** NIF version is **7**. The local server has sent three messages to the server g_bombay and it has received two messages from g_bombay.

```
IBM Informix Dynamic Server Version 10.00.UC1  -- On-Line -- Up 00:02:34 -- 28672 Kbytes
NIF anchor Block: af01610
       nifGState        RUN
          RetryTimeout   300

CDR connections:
 Id    Name       State    Version    Sent   Received
 -----------------------------------------------------
 200 g_bombay      RUN        7         3        2
```

Figure 14-39. onstat -g nif Output

The onstat -g nsc client_id Option

If no *client_id* is provided, information about all current shared memory connections to the database server is given. If a *client_id* is provided then this command gives more detailed information about the shared memory connection with that ID.

Example Output:

This is output of **onstat -g nsc** with no *client_id*. It shows that there is only one user currently connecting to the database server through shared memory. That connection has an ID of 0.

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 6 days

  clientid  clientPID      state #serverbufs #clientbufs    #rdwrts
         0       6031  Connected           4           4         12
```

Figure 14-40. onstat -g nsc Output

This example shows output from running the command using a *client_id* of 0.

```
Network Shared Memory Status for Client: 0

    clientid    clientPID        state  #serverbufs  #clientbufs      #rdwrts
           0        18949    Connected            4            4       447048

     needbuf        segid        semid       semnum     be_semid    be_semnum
           0         1303       851969            0       851969           10

  be_curread  be_curwrite   fe_curread  fe_curwrite
          -1            1            0            2

 be_nextread be_nextwrite  fe_nextread fe_nextwrite
           2            2            4            3

readyqueue
  -1   -1   -1   -1   -1   -1   -1   -1   -1   -1   -1

     Server Buffers                         Client Buffers
  i: bufid     status   offset  fe_addr        bufid     status   offset  fe_addr
  0:     4      inuse     4474   804474            0      avail     3424   803424
  1:     5      inuse     4888   804888            1      avail     3838   803838
  2:     6      avail     4c9c   804c9c            2      inuse     3c4c   803c4c
  3:     7      avail     50b0   8050b0            3      avail     4060   804060
  4:    -1       free        0        0           -1       free        0        0
  5:    -1       free        0        0           -1       free        0        0
```

Figure 14-41. onstat -g nsc with client id Output

Output Description:

clientid
: Server assigned ID

clientPID
: Client process ID

state
: State of connection

  Connected
  : The client has established a connection with the server.

  Con1
  : The server has successfully set up a connection with the client, but the client has not yet been notified of it.

  Waiting
  : The server is in the process of setting up a connection with the client.

  Reject
  : Client connection has been rejected by the server, normally because the server is shutting down or not yet in on-line mode.

  Closed
  : Server has closed the connection with the client. Client might not be aware of the fact yet.

  Not connected
  : Server is initializing internal structures for the connection.

  Unknown
  : Connection has been closed and the client is aware of the fact. Server is cleaning up internal structures.

#serverbufs
: Database server buffers currently allocated

#clientbufs
: Client buffers currently allocated

#rdwrts
: The total number of reads and writes performed through this connection since it was created.

The following items are only in the output if you run **onstat -g nsc** with a *client_id*:

needbuf
: Indicates if server is waiting for a buffer to be freed

  0 False

  1 True

segid
: Shared memory segment ID

semid
: Semaphore ID

semnum
: Semaphore number in the semaphore ID

be_semid
: Backend semaphore ID

be_semnum
: Backend semaphore number in the semaphore ID

be_curread
: ID of backend buffer being read

be_curwrite
: ID of backend buffer being written

fe_curread
: ID of frontend buffer being read

fe_currwrite
: ID of frontend buffer being written

be_nextread
: ID of next backend buffer to be read

be_nextwrite
: ID of next backend buffer to be written

fe_nextread
: ID of next frontend buffer to be read

fe_nextwrite
: ID of next frontend buffer to be written

readyqueue
: Queue of the shared memory buffer ids

Buffers

i
: Internal location key of message buffer

bufid
: Message buffer ID

status
: Status of message buffer

offset
: Offset of memory buffer in shared memory segments

fe_addr
: Frontend address of message buffer

The onstat -g nsd Option

The **onstat -g nsd** option prints shared-memory data for poll threads.

Example Output:

```
Network Shared Memory Data for Poll Thread: 0
Free Message Buffer Bitmap
(bitmap address = 10b9eef80, bitmap size 480)
000000010b9eef80:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
000000010b9eefa0:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
Free Message Buffer Status Bitmap
(bitmap address = 10ca0a9b0, bitmap size 50)
000000010ca0a9b0:ffffffff ffffff
Message Buffer Table
bufid   clientid            addr
Message Buffer Status Table
clientid      netscb addr            addr           offset
```

Figure 14-42. onstat -g nsd Output

The onstat -g ntd Option

The **onstat -g ntd** option prints network statistics by service.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:57:24 -- 101376 Kbytes
global network information:
  #netscb connects     read    write    q-free  q-limits  q-exceed alloc/max
   3/   3        0        0        0    1/   1  135/  10    0/   0    2/   0
Client Type     Calls   Accepted   Rejected       Read      Write
sqlexec         yes            6          0          0          0
srvinfx         yes            0          0          0          0
onspace         yes            0          0          0          0
onlog           yes            0          0          0          0
onparam         yes            0          0          0          0
oncheck         yes            0          0          0          0
onmonitor       yes            0          0          0          0
dr_accept       yes            0          0          0          0
ontape          yes            0          0          0          0
srvstat         yes            0          0          0          0
asfecho         yes            0          0          0          0
listener        yes            0          0          0          0
crsamexec       yes            0          0          0          0
onutil          yes            0          0          0          0
Totals                         6          0          0          0
```

Figure 14-43. onstat -g ntd Output

The onstat -g ntm Option

The **onstat -g ntm** option prints network mail statistics.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:57:33 -- 101376 Kbytes
global network information:
  #netscb connects     read    write    q-free  q-limits  q-exceed alloc/max
   3/   3       0       0       0    1/   1  135/  10    0/   0    2/   0
Network mailbox information:
 box       netscb   thread name     max received   in box   max in box full signal
   5      c631028  tlitcppoll     10         0      0         0       0    yes
   6      c63e548  tlitcplst      10         0      0         0       0     no
```

Figure 14-44. onstat -g ntm Output

The onstat -g ntt Option

The **onstat -g ntt** option prints network user times.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:57:44 -- 101376 Kbytes
global network information:
  #netscb connects     read    write    q-free  q-limits  q-exceed alloc/max
   3/   3       0       0       0    1/   1  135/  10    0/   0    2/   0
Individual thread network information (times):
 netscb thread name sid  open     read     write     address
c76ea28 ontape      61   14:34:48 14:34:50 14:34:50
c63e548 tlitcplst    4   14:30:43 14:34:48           server.ibm.com|5006|tlitcp
c631028 tlitcppoll   3   14:32:32
```

Figure 14-45. onstat -g ntt output

The onstat -g ntu Option

The **onstat -g ntu** option prints network user statistics.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:57:53 -- 101376 Kbytes
global network information:
  #netscb connects     read    write    q-free  q-limits  q-exceed alloc/max
   3/   3       0       0       0    1/   1  135/  10    0/   0    2/   0
Individual thread network information (basic):
   netscb type   thread name    sid   fd poll    reads   writes q-nrm q-pvt q-exp
  c76ea28 tlitcp ontape          61    3    5        0        0  0/ 0  1/ 1  0/ 0
  c63e548 tlitcp tlitcplst        4    1    5        0        0  0/ 0  0/ 0  0/ 0
  c631028 tlitcp tlitcppoll       3    0    5        0        0  0/ 0  0/ 0  0/ 0
```

Figure 14-46. onstat -g ntu Output

The onstat -g pos Option

The **onstat -g pos** option prints the values for the **$INFORMIXDIR/etc/.infos.DBSERVERNAME** file.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:58:04 -- 101376 Kbytes
  1   7   0 infos ver/size 6 520
  2   1   0 snum=  101 shmk=52665801 shmb=000000000a000000 cosvr=1 gpid=2599 qa10_1
  3   4   0 onconfig path /work/xps/sqldist/etc/onconfig.xps
  4   5   0 host qa10-1
  5   6   0 oninit ver IBM Informix Extended Parallel Server Version 8.50.FN145
  6   8   0 infos sqlhosts: /work/xps/sqldist/etc/sqlhosts
  7  12   0 del
  8  13   0 del
  9   2 4001 shm  id=4001  key=0x52665801 (1382438913) addr=0x    a000000 size=19918848   R
 10   3   1 sema     1
 11  11   0 MRI: addr = 0xb4110e8 version = 0x10001
 12   2   2 shm  id=2     key=0xab00bf7c (-1426014340) addr=0x  200000000 size=16777216   R
 13   2   3 shm  id=3     key=0x52665802 (1382438914) addr=0x     bb00000 size=9437184    V
 14   2   5 shm  id=5     key=0x52665803 (1382438915) addr=0x     c400000 size=8388608    V
 15   2   7 shm  id=7     key=0x52665804 (1382438916) addr=0x     cc00000 size=32505856   V
 16   2   8 shm  id=8     key=0x52665805 (1382438917) addr=0x     eb00000 size=8388608    V
 17   2   9 shm  id=9     key=0x52665806 (1382438918) addr=0x     f300000 size=8388608    V
 18   2  10 shm  id=10    key=0x52665807 (1382438919) addr=0x     fb00000 size=8388608    V
```

Figure 14-47. onstat -g pos Output

The onstat -g ppf partition number | 0 Option

The **onstat -g ppf** option prints the partition profile for ***partition number***; 0 prints profiles for all partitions. If TBLSPACE_STATS configuration parameter is set to 0, displays: Partition profiles disabled.

Example Output:

```
Partition profiles
partnum    lkrqs lkwts dlks  touts isrd  iswrt isrwt isdel bfrd  bfwrt seqsc rhitratio
0x100001   0     0     0     0     0     0     0     0     0     0     0     0
0x100002   1506  0     0     0     416   4     0     4     1282  20    0     97
0x100003   15    0     0     0     5     0     0     0     20    0     0     75
0x1000a5   0     0     0     0     0     0     0     0     12    0     0     67
0x1000e3   4     0     0     0     1     0     0     0     4     0     0     25
0x200001   0     0     0     0     0     0     0     0     0     0     0     0
0x300001   0     0     0     0     0     0     0     0     0     0     0     0
0x400001   0     0     0     0     0     0     0     0     0     0     0     0
```

Figure 14-48. onstat -g ppf Output

Output Description:

partnum
Partition number

lkrqs
Lock requests

lkwts
Lock waits

dlks
Deadlocks

touts
Remote deadlock timeout

isrd
Number of reads

iswrt
Number of rewrites

isdel
Deletes

bfrd Buffer reads

bfwrt Buffer writes

seqsc Sequential scans

rhitratio Ration of disk read to buffer read

The onstat -g prc Option

The **onstat -g prc** option prints the number of sessions currently using the UDR or SPL routine.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:59:00 -- 101376 Kbytes
Stored Procedure Cache:
    Number of lists        : 31
    PC_POOLSIZE            : 50
    Number of entries        : 0
    Number of inuse entries    : 0
Stored Procedure Cache Entries:
list#  id  ref_cnt   dropped?   heap_ptr     procedure name
---------------------------------------------------------------
Stored Procedure Cache is empty.
```

Figure 14-49. onstat -g prc Output

The onstat -g que Option

The **onstat -g que** command prints statistics that are common to all queues in Enterprise Replication. The queuer manages the logical aspects of the queue. The RQM (reliable queue manager) manages the physical queue.

The **-g que** option is used primarily as a debugging tool and by Technical Support.

Example Output:

In the following example, **Element high water mark** shows the maximum size of the transaction buffer header data (metadata) allowed in memory, shown in kilobytes. **Data high water mark** shows the maximum size of transactions for user data allowed in memory, shown in kilobytes.

```
IBM Informix Dynamic Server Version 10.00.UC1   -- On-Line -- Up 00:40:28 -- 28672 Kbytes
CDR Queuer Statistics:
  Queuer state           : 2
  Local server           : 100
  Element high water mark : 131072
  Data high water mark    : 131072
  # of times txns split   : 0
  Total # of split txns   : 0
  allowed log delta       : 30
  maximum delta detected  : 4
  Control Key             : 0/00000007
  Synchronization Key     : 0/00000003
Replay Table:
  Replay Posn (Disk value): 12/00000018 (12/00000018)
  Replay save interval    : 10
  Replay updates          : 10
  Replay # saves          : 17
  Replay last save time   : (1095118157) 2004/09/13 18:29:17
Send Handles
  Server ID               : 200
  Send state,count        : 0,0
  RQM hdl for trg_send:  Traverse handle (0xaf8e018) for thread CDRACK_0 at Head_of_Q,
      Flags: None
  RQM hdl for control_send:  Traverse handle (0xaf74018)
    for thread CDRACK_0 at Head_of_Q,  Flags: None
  RQM hdl for sync_send:  Traverse handle (0xadc6018) for thread CDRACK_0 at Head_of_Q,
      Flags: None
  Server ID               : 200
  Send state,count        : 0,0
  RQM hdl for trg_send:  Traverse handle (0xac8b018) for thread CDRACK_1 at Head_of_Q,
      Flags: None
  RQM hdl for control_send:  Traverse handle (0xb1ce018) for thread CDRACK_1 at Head_of_Q,
      Flags: None
  RQM hdl for sync_send:  Traverse handle (0xadc5018) for thread CDRACK_1 at Head_of_Q,
      Flags: None
  Server ID               : 200
  Send state,count        : 0,0
  RQM hdl for trg_send:  Traverse handle (0xaea71d8) for thread CDRNsA200 at Head_of_Q,
      Flags: None
  RQM hdl for ack_send:  Traverse handle (0xae8c1d8) for thread CDRNsA200 at Head_of_Q,
      Flags: None
  RQM hdl for control_send:  Traverse handle (0xae9e1d8) for thread CDRNsA200 at Head_of_Q,
      Flags: None
```

Figure 14-50. onstat -g que Output

The onstat -g rbm Option

The **onstat -g rbm** option prints block bit map for the resident segment (communication message area).

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:59:23 -- 101376 Kbytes
Block bitmap for resident segment address 0xa000000:
 address = 0xa000290, size(bits) = 4863
 used = 4820, largest_free = -1
 firstfree = 1, lastalloc=4820
      0:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
    256:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
    512:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
    768:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   1024:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   1280:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   1536:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   1792:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   2048:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   2304:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   2560:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   2816:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   3072:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   3328:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   3584:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   3840:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   4096:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   4352:ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
   4608:ffffffffffffffff ffffffffffffffff ffffffffffffffff fffff00000000000
```

Figure 14-51. onstat -g rbm Output

The onstat -g rcv Option

The **onstat -g rcv** command prints statistics about the receive manager in Enterprise Replication. The receive manager is a set of service routines between the receive queues and data sync.

The **onstat -g rcv** command has the following formats:

```
onstat -g rcv
onstat -g rcv serverid
onstat -g rcv full
```

The *serverID* modifier causes the command to print only those output messages received from the replication server whose groupID is *serverid*. The *full* modifier causes the command to print all statistics.

The **onstat -g rcv** command includes the Receive Manager global section. In this section, the following fields have the meanings shown:

Field	Description
cdrRM_DSParallelPL	Shows the current level of Apply Parallelism, 0 (zero) being the highest
cdrRM_DSNumLockTimeout cdrRM_DSNumLockRB cdrRM_DSNumDeadLocks	Indicate the number of collisions between various apply threads
cdrRM_acksinList	Shows acknowledgements that have been received but not yet processed

The **onstat -g rcv** command includes the Receive Parallelism Statistics section, a summary of the data sync threads by source server.

Field	Description
Server	Source server ID
Tot.Txn.	Total number of transactions applied from this source server
Pending	Number of current transactions in the pending list for this source server
Active	Number of current transactions currently being applied from this source server
MaxPnd	Maximum number of transactions in the pending list queue
MaxAct	Maximum number of transaction in the active list queue
AvgPnd	Average depth of the pending list queue
AvgAct	Average depth of the active list queue
CommitRt	Commit rate of transaction from this source server based on transactions per second

The Statistics by Source section of the **onstat -g rcv** command shows the following information for each source server. For each replicate ID:

- The number of transactions applied from the source servers
- The number of inserts, deletes, and updates within the applied transactions
- The timestamp of the most recently applied transaction on the target server
- The timestamp of the commit on the source server for the most recently applied transaction

The **-g rcv** option is used primarily as a debugging tool and by Technical Support. If you suspect that acknowledgement messages are not being applied, you can use this option to check.

Example Output:

The following example shows output for the **onstat -g rcv full** command.

```
Receive Manager global  block 0D452018
    cdrRM_inst_ct:                      5
    cdrRM_State:              00000000
    cdrRM_numSleepers:        3
    cdrRM_DsCreated:          3
    cdrRM_MinDSThreads:       1
    cdrRM_MaxDSThreads:       4
    cdrRM_DSBlock             0
    cdrRM_DSParallelPL        0
    cdrRM_DSFailRate          0.000000
    cdrRM_DSNumRun:           35
    cdrRM_DSNumLockTimeout    0
    cdrRM_DSNumLockRB         0
    cdrRM_DSNumDeadLocks      0
    cdrRM_DSNumPCommits       0
    cdrRM_ACKwaiting          0
    cdrRM_totSleep:           77
    cdrRM_Sleeptime:          153
    cdrRM_Workload:           0
    cdrRM_optscale:           4
    cdrRM_MinFloatThreads:    2
    cdrRM_MaxFloatThreads:    7
    cdrRM_AckThreadCount:     2
    cdrRM_AckWaiters:         2
    cdrRM_AckCreateStamp:Wed Sep 08 11:47:49 2004
    cdrRM_DSCreateStamp: Wed Sep 08 14:16:35 2004
    cdrRM_acksInList:         0
    cdrRM_BlobErrorBufs:      0
```

Figure 14-52. onstat -g rcv Output (Part 1 of 2)

```
Receive Parallelism Statistics
Srvr Tot.Txn. Pnding Active MaxPnd MaxAct AvgPnd AvgAct CommitRt
   1     35        0      0     21      3   7.00   1.63    0.00
   5      3        0      0      1      1   1.00   1.00    0.02
   6      6        0      0      1      1   1.00   1.00    0.21
Tot Pending:0 Tot Active:0 Avg Pending:5.77 Avg Active:1.50
Commit Rate:0.01

Time Spent In RM Parallel Pipeline Levels
Lev. TimeInSec  Pcnt.
   0     17405 100.00%
   1         0   0.00%
   2         0   0.00%

Statistics by Source
Server 1
Repl  Txn Ins Del Upd Last Target Apply   Last Source Commit
65541  23   0   1 616 2004/09/08 14:20:15 2004/09/08 14:20:15
65542  11   0   0 253 2004/09/08 14:19:33 2004/09/08 14:19:33
65545   1   0  67   0 2004/09/08 14:20:37 2004/09/08 14:20:37
Server 5
Repl  Txn Ins Del Upd Last Target Apply   Last Source Commit
65541   3   0   0  81 2004/09/08 16:36:10 2004/09/08 16:36:09
Server 6
Repl  Txn Ins Del Upd Last Target Apply   Last Source Commit
65548   6   0   0  42 2004/09/08 16:37:59 2004/09/08 16:37:58
```

Figure 14-52. onstat -g rcv Output (Part 2 of 2)

The onstat -g rea Option

The **onstat -g rea** option prints ready threads.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 01:59:34 -- 101376 Kbytes

Ready threads:
 tid     tcb     rstcb      prty status       vp-class      #scheds name
```

Figure 14-53. onstat -g rea Output

The onstat -g rep Option

The **onstat -g rep** command prints events that are in the queue for the schedule manager for Enterprise Replication. The **-g rep** option is used primarily as a debugging tool and by Technical Support.

The **onstat -g rep** command has the following formats:

```
onstat -g rep
onstat -g rep replname
```

The *repl_name* modifier limits the output to those events originated by the replicate named *repl_name*.

Example Output:

The following example shows sample output for the **onstat -g rep** command.

```
IBM Informix Dynamic Server Version 10.00.UC1  -- On-Line -- Up 00:30:10 -- 28672 Kbytes
Schedule manager Cb: add7e18 State: 0x8100 <CDRINIT,CDRRUNNING>
Event        Thread            When
============================================
CDRDS        CDREvent        00:00:20
```

Figure 14-54. onstat -g rep Output

The onstat -g rqm Option

The **onstat -g rqm** command prints statistics and contents of the low-level queues (send queue, receive queue, ack send queue, sync send queue, and control send queue) managed by the Reliable Queue Manager (RQM) in Enterprise Replication. The RQM manages the insertion and removal of items to and from the various queues. The RQM also manages spooling of the in-memory portions of the queue to and from disk. The **-g rqm** option displays the contents of the queue, size of the transactions in the queue, how much of the queue is in memory and on disk, the location of various handles to the queue, and the contents of the various progress tables. You can choose to print information for all queues or for just one queue by using one of the modifiers described below.

If a queue is empty, no information is printed for that queue.

The **onstat -g rqm** command has the following formats:

```
onstat -g rqm
onstat -g rqm modifier
```

The following table describes the values for *modifier*.

Modifier	Action
ACKQ	Prints the ack send queue
CNTRLQ	Prints the control send queue
RECVQ	Prints the receive queue
SENDQ	Prints the send queue
SYNCQ	Prints the sync send queue
FULL	Prints full information about every in-memory transaction for every queue
BRIEF	Prints a brief summary of the number of transactions in each of the queues and the replication servers for which the data is queued Use this modifier to quickly identify sites where a problem exists. If large amounts of data are queued for a single server, then that server is probably down or off the network.
VERBOSE	Prints all the buffer headers in memory

When you specify a modifier to select a specific queue, the command prints all the statistics for that queue and information about the first and last in-memory transactions for that queue.

The other modifiers of the **onstat -g rqm** command are used primarily as a debugging tool and by Technical Support.

The output for the SENDQ modifier contains the following sections:

- RQM Statistics for Queue—a summary of current and historical information for the queue. This includes the number of transactions in the queue, how many are spooled, how many bytes they are using, some maximum statistics, and the high water marks that will trigger stably storing transactions in the **syscdr** tables.
- First Txn—information about the first transaction in the queue. To check if the queue is draining, you can run **onstat -g rqm** several times and see if the first transaction's RQM key is changing. The RQM key has the following format: *Server_ID*/*Commit_unique_logID*/*Commit_log_position*/*Sequence*. If it is not draining, the target server may be offline or some other problem is occurring. The NeedAck field shows from which server the transaction is waiting for an acknowledgement. You can use this bitmap mask with the output from the **onstat -g cat** command to determine the name of the server which server Enterprise Replication is waiting on for an acknowledgement.
- Last Txn—information about the last transaction in the queue
- Traverse handle—lists the handles used for threads
- Progress table—provides information about the progress of each replicate under the headers: Server, Group, Bytes Queued, Acked, and Sent. The Group field shows the replicate ID. The Acked field shows what has been acknowledged. The Sent field shows which entries are now in transit. Both the Acked and the Sent field show the RQM key, which has the following format: *Server_ID*/*Commit_unique_logID*/*Commit_log_position*/*Sequence*.

Example Output:

The following example shows output for the **onstat -g rqm SENDQ** command.

```
RQM Statistics for Queue (0x0D3DF018) trg_send
 Transaction Spool Name: trg_send_stxn
 Insert Stamp: 35/0
 Flags: SEND_Q, SPOOLED, PROGRESS_TABLE, NEED_ACK
 Txns in queue:             35
 Log Events in queue:       0
 Txns in memory:            35
 Txns in spool only:        0
 Txns spooled:              0
 Unspooled bytes:           176206
 Size of Data in queue:     176206 Bytes
 Real memory in use:        176206 Bytes
 Pending Txn Buffers:       0
 Pending Txn Data:          0 Bytes
 Max Real memory data used: 176206 (2457600) Bytes
 Max Real memory hdrs used  65988 (2457600) Bytes
 Total data queued:         176206 Bytes
 Total Txns queued:         35
 Total Txns spooled:        0
 Total Txns restored:       0
 Total Txns recovered:      0
 Spool Rows read:           0
 Total Txns deleted:        0
 Total Txns duplicated:     0
 Total Txn Lookups:         363
```

Figure 14-55. onstat -g rqm SENDQ Output (Part 1 of 3)

```
First Txn (0x0D60C018) Key:  1/9/0x000d4bb0/0x00000000
 Txn Stamp: 1/0, Reference Count: 0.
 Txn Flags: Notify
 Txn Commit Time: (1094670993) 2004/09/08 14:16:33
 Txn Size in Queue: 5908
 First Buf's (0x0D31C9E8) Queue Flags: Resident
 First Buf's Buffer Flags: TRG, Stream
 NeedAck: Waiting for Acks from <[0004]>
 No open handles on txn.

 Last Txn (0x0D93A098) Key:  1/9/0x00138ad8/0x00000000
 Txn Stamp: 35/0, Reference Count: 0.
 Txn Flags: Notify
 Txn Commit Time: (1094671237) 2004/09/08 14:20:37
 Txn Size in Queue: 6298
 First Buf's (0x0D92FFA0) Queue Flags: Resident
 First Buf's Buffer Flags: TRG, Stream
 NeedAck: Waiting for Acks from <[0004]>
```

Figure 14-55. onstat -g rqm SENDQ Output (Part 2 of 3)

```
 Traverse handle (0x0D045018) for thread CDRNsA3 at txn (0x0D93A098) End_of_Q,Flags: None
 Traverse handle (0x0D08E018) for thread CDRNsA4 at txn (0x0D93A098) End_of_Q,Flags: None
 Traverse handle (0x0D523018) for thread CDRNsA5 at txn (0x0D93A098) End_of_Q,Flags: None
 Traverse handle (0x0D0D9018) for thread CDRNsA6 at txn (0x0D93A098) End_of_Q,Flags: None

 Traverse handle (0x0D4041D8) for thread CDRNsA2 at Head_of_Q,Flags: None
 Traverse handle (0x0D3F01D8) for thread CDRNrA2 at Head_of_Q,  Flags: None
 Traverse handle (0x0D045018) for thread CDRNsA3 at txn (0x0D93A098) End_of_Q,Flags: None
 Traverse handle (0x0D31C018) for thread CDRNrA3 at Head_of_Q,  Flags: None
 Traverse handle (0x0D08E018) for thread CDRNsA4 at txn (0x0D93A098) End_of_Q,Flags: None
 Traverse handle (0x0D4C8018) for thread CDRNrA4 at Head_of_Q,  Flags: None
 Traverse handle (0x0D523018) for thread CDRNsA5 at txn (0x0D93A098) End_of_Q,Flags: None
 Traverse handle (0x0D57F018) for thread CDRNrA5 at Head_of_Q,  Flags: None
 Traverse handle (0x0D0D9018) for thread CDRNsA6 at txn (0x0D93A098) End_of_Q,Flags: None

Server    Group Bytes Queued      Acked    Sent
-----------------------------------------------------------
    6  0x10009          0 1/9/138ad8/0   -    1/9/138ad8/0
    5  0x10009          0 1/9/138ad8/0   -    1/9/138ad8/0
    4  0x10009          0 1/9/138ad8/0   -    1/9/138ad8/0
    3  0x10009          0 1/9/138ad8/0   -    1/9/138ad8/0
    2  0x10009       4154 efffffff/efffffff/efffffff/efffffff - 1//138ad8/0
    6  0x10006          0 1/9/12d8f8/0   -    1//12d8f8/0
    5  0x10006          0 1/9/12d8f8/0   -    1//12d8f8/0
    4  0x10006          0 1/9/12d8f8/0   -    1/9/12d8f8/0
    3  0x10006          0 1/9/12d8f8/0   -    1/9/12d8f8/0
    2  0x10006       31625 efffffff/efffffff/efffffff/efffffff – 1/9/12d8f8/0
```

Figure 14-55. onstat -g rqm SENDQ Output (Part 3 of 3)

The onstat -g rwm Option

Prints read and write mutexes.

Example Output:

```
MUTEX   NAME     write/read/wait    tcb list
<address> <name>     first mutex
       Writer     ticket = <ticket address>   tcb=<thread address> <thread name>
       Readers    ticket = <ticket address>   tcb=<thread address> <thread name>
       Waiters    ticket = <ticket address>   tcb=<thread address> <thread name>
<address> <name>        second mutex
       Writer     ticket = <ticket address>   tcb=<thread address> <thread name>
       Readers     ticket = <ticket address>  tcb=<thread address> <thread name>
       Waiters     ticket = <ticket address>  tcb=<thread address> <thread name>
....
....
....
<address> <name>       last mutex
       Writer     ticket = <ticket address>   tcb=<thread address> <thread name>
       Readers    ticket = <ticket address>   tcb=<thread address> <thread name>
       Waiters    ticket = <ticket address>   tcb=<thread address> <thread name>
```

Figure 14-56. onstat -g rwm Output

Output Description:

tcb List of thread addresses

Writer List of write threads

Readers
: List of read threads

Waiters
: List of waiting threads

ticket
: Address of ticket acquired by the thread

The onstat -g sch Option

The **onstat -g sch** option prints the number of semaphore operations, spins, and busy waits for each virtual processor.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 02:00:03 -- 101376 Kbytes
VP Scheduler Statistics:
 vp    pid       class       semops    busy waits  spins/wait
 1     2599      cpu         0         0           0
 2     2602      adm         0         0           0
 3     2603      cpu         125735    125735      10001
 4     2604      cpu         125486    125487      10291
 5     2605      cpu         125585    125585      10001
 6     2606      lio         811       811         1000      ]
 7     2607      pio         810       810         1000
 8     2608      aio         1489      1489        1000
 9     2609      msc         810       810         1000
 10    2610      fifo        810       810         1000
 11    2611      fifo        811       811         1000
 12    2612      aio         812       812         1000
 13    2613      aio         810       810         1000
 14    2614      aio         811       811         1000
Thread Migration Statistics:
 vp   pid     class steal-at steal-sc idlvp-at idlevp-sc Q-ln Polls Idles IdleSec
 1    2599    cpu   125799   184      0        0        0      151   151    2.99
 2    2602    adm   0        0        186      186      0        0     0    0.00
 3    2603    cpu   125596   43       1        0        0        0     0    0.00
 4    2604    cpu   125481   8        2        0        0        0     0    0.00
 5    2605    cpu   125594   17       0        0        0        0     0    0.00
 6    2606    lio   0        0        0        0        0        0     0    0.00
 7    2607    pio   0        0        0        0        0        0     0    0.00
 8    2608    aio   0        0        3        0        0        0     0    0.00
 9    2609    msc   0        0        0        0        0        0     0    0.00
 10   2610    fifo  0        0        0        0        0        0     0    0.00
 11   2611    fifo  0        0        0        0        0        0     0    0.00
 12   2612    aio   0        0        0        0        0        0     0    0.00
 13   2613    aio   0        0        0        0        0        0     0    0.00
 14   2614    aio   0        0        0        0        0        0     0    0.00
```

Figure 14-57. onstat -g sch Output

The onstat -g seg Option

The **onstat -g seg** option prints shared-memory segment statistics. This option shows how many segments are attached and their sizes.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 02:00:13 -- 101376 Kbytes
Segment Summary:
id        key         addr     size            ovhd     class blkused  blkfree
4001      1382438913  a000000    19918848       1760     R     4820     43
(shared)  1382438913  b2ff000    8392704        928      V     2049     0
3         1382438914  bb00000    9437184        952      V     2304     0
5         1382438915  c400000    8388608        920      V     1724     324
7         1382438916  cc00000   32505856       1656      V     7936     0
8         1382438917  eb00000    8388608        920      V     282      1766
9         1382438918  f300000    8388608        920      V     393      1655
10        1382438919  fb00000    8388608        920      V     393      1655
Total:    -           -        103809024        -        -     19901    5443
   (* segment locked in memory)
```

Figure 14-58. onstat -g seg Output

The onstat -g ses Option

The **onstat -g ses** option prints session-related information. You can specify one of the following invocations.

Invocation	Explanation
onstat -g ses	Displays a one-line summary for each session
onstat -g ses *sessionid*	Displays information for a specific session

Figure 14-59 shows the output of the **onstat -g ses** option.

```
onstat -g ses

IBM Informix Dynamic Server Version 10.00.UC3     -- On-Line -- Up 7 days 18:43:13 --
 38912 Kbytes

session                                       #RSAM    total      used       dynamic
id       user     tty      pid      hostname threads  memory     memory     explain
24       informix -        0        -        0        12288      7936       off
23       informix -        17602    carson   1        57344      48968      off
3        informix -        0        -        0        12288      9168       off
2        informix -        0        -        0        12288      7936       off
```

Figure 14-59. onstat -g ses Output

Figure 14-60 shows the output of the **onstat -g ses** *sessionid* option.

```
IBM Informix Dynamic Server Version 10.00.UC3     -- On-Line -- Up 2 days 19:42:
11 -- 38912 Kbytes

session                                      #RSAM    total      used       dynamic
id       user     tty      pid      hostname threads  memory     memory     explain
16       sitaramv 1        18523    carson   1        81920      71720      off

tid      name     rstcb    flags    curstk   status
35       sqlexec  a7ed9f4  Y--P---  5488     cond wait(netnorm)

Memory pools    count 1
name         class addr     totalsize freesize #allocfrag #freefrag
16           V     afea020  81920     10200    119        13

name           free       used           name           free       used
overhead       0          1648           scb            0          96
opentable      0          1768           filetable      0          336
log            0          21880          temprec        0          16200
keys           0          680            ralloc         0          5120
gentcb         0          1208           ostcb          0          2528
sqscb          0          13216          sql            0          40
rdahead        0          184            hashfiletab    0          280
osenv          0          1920           sqtcb          0          2024
fragman        0          208            udr            0          312
xatm           0          2072

sqscb info
scb      sqscb    optofc   pdqpriority sqlstats optcompind  directives
adff580  af93018  0        0           0        2           1

Sess  SQL            Current            Iso Lock       SQL  ISAM F.E.
Id    Stmt type      Database           Lvl Mode       ERR  ERR  Vers Explain
16    -              xabasicdb          RR  Not Wait   0    0    9.03 Off


Last parsed SQL statement :
  EXECUTE FUNCTION xa2pc_mi_unregister("xads_t2_i2")

Xadatasources participated in this session :
Xadatasource name                                     RMID       Active
xabasicdb@atmol10:sitaramv.xads_t3_i1                    6        YES
xabasicdb@atmol10:sitaramv.xads_t2_i1                    4        YES
xabasicdb@atmol10:sitaramv.xads_t1_i3                    3        YES
xabasicdb@atmol10:sitaramv.xads_t1_i2                    2        YES
xabasicdb@atmol10:sitaramv.xads_t1_i1                    1        YES
xabasicdb@atmol10:sitaramv.xads_t2_i2                    5        NO
```

Figure 14-60. onstat -g ses sessionid Output

You can interpret the output from this option as follows:

Session section

Session id	The session ID
user	The username who started the session
tty	The tty associated with the front-end for this session
pid	The process ID associated with the front-end for this session
hostname	The hostname from which this session has connected
#RSAM threads	The number of RSAM thread allocated for this session
total memory	The amount of memory allocated for this session

used memory The amount of memory actually used by this session

dynamic explain
Generate explain output of the sql statements of the session (`on` or `off`)

Threads section

tid The thread ID

name The name of the thread

rstcb RSAM control block

flags Describes the status of the thread using the following codes:

Position 1

B Waiting on a buffer

C Waiting on a checkpoint

G Waiting on a logical-log buffer write

L Waiting on a lock

S Waiting on a mutex

T Waiting on a transaction

X Waiting on a transaction cleanup

Y Waiting on a condition

Position 2

* An asterisk in this position means that the thread encountered an I/O failure in the middle of a transaction

Position 3

A Archive thread

B Begin work

P Begin Prepare or Prepared work

X XA prepared

C Committing or committed

R Aborting or aborted

H Heuristically aborted or heuristically rolling back

Position 4

P Primary thread

Position 5

R Reading

X Critical section

Position 6

R Recovery thread

Position 7

M Monitor thread

D	Daemon thread
C	Cleaner
F	Flusher
B	btcleaner

curstk Current stack size

status Current thread status

Memory pools header section. The information is repeated for each session pool.

name	Name of pool
class	Class of the memory where the pool is allocated from. R is for Resident, V is for Virtual, and M is for Message
addr	Address of the pool structure
totalsize	Total size of the memory acquired by the pool in bytes
freesize	Number of bytes free in the pool
#allocfrag	Number of allocated memory fragments in the pool
#freefrag	Number of free fragments in the pool

The memory pool section

name	Name of a component which has allocated memory from the pool
free	Number of bytes freed
used	Number of bytes allocated

The sqscb information section

scb	The session control block. This is the address of the main session structure in shared memory.
sqscb	SQL level control block of the session
optofc	The current value of the **OPTOFC** environment variable or onconfig setting.
pdqpriority	The current value of the **PDQPRIORITY** environment variable or onconfig setting.
sqlstats	The current value of the **SQLSTATS** environment variable or onconfig setting.
optcompind	The current value of the **OPTCOMPIND** environment variable or onconfig setting.
directives	The current value of the **DIRECTIVES** environment variable or onconfig setting.

The Last parsed SQL statement section has the same information as the **onstat -g sql** option. See "The onstat -g sql Option" on page 14-71.

Xadatasources participated in this session section shows information about the XA data sources that are available during the session, their resource manager identifiers, and whether they are currently active.

Xdatasource name
The XA data source that participated in the session

RMID The identifier of the resource manager for the corresponding XA data source

Active Whether the XA data source is still active

The onstat -g sle Option

The **onstat -g sle** option prints all sleeping threads.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 02:00:27 -- 101376 Kbytes
Current Admin VP sleep period:  10  millisecs
Sleeping threads with timeouts: 21  threads
   tid v_proc           rstcb            name     time
    49      1         b3b13a8       onmode_mon     0.02
     5      1               0  Cosvr Avail Mgr     0.05
    42      1         b3ad028      main_loop()     0.08
     9      3         b3ad6e8         xtm_svcc     0.64
    14      5               0       mgmt_thd_5     0.65
    13      4               0       mgmt_thd_4     0.65
     4      1               0       mgmt_thd_1     0.65
     6      3               0          dfm_svc     0.98
    33     13               0      mgmt_thd_13     1.54
    27     10               0      mgmt_thd_10     1.54
    21      7               0       mgmt_thd_7     1.54
    12      3               0       mgmt_thd_3     1.76
    29     11               0      mgmt_thd_11     1.76
    23      8               0       mgmt_thd_8     2.08
    31     12               0      mgmt_thd_12     2.08
    35     14               0      mgmt_thd_14     2.98
    19      6               0       mgmt_thd_6     3.00
    25      9               0       mgmt_thd_9     3.00
    37      3               0          sch_rgm     3.48
    44      5         b3af8a8      btscanner 0     7.31
    46      3         b3b0628        bum_sched    41.26
```

Figure 14-61. onstat -g sle Output

The onstat -g sql Option

The **onstat -g sql** option prints SQL-related information about a session. You can specify one of the following invocations.

Invocation	Explanation
onstat -g sql	Displays a one line summary for each session
onstat -g sql ***sessionid***	Displays SQL information for a specific session

Note: Encrypted passwords and password hint parameters in encryption functions are not shown. Figure 14-62 on page 14-72 displays an encrypted password in the `Last parsed SQL statement` field.

Figure 14-62 shows the output of the **onstat -g sql** option.

```
onstat -g sql 22
IBM Informix Dynamic Server Version 10.00.UC1     -- On-Line -- Up 00:07:38 -- 19456 Kbytes
Sess  SQL           Current            Iso Lock       SQL  ISAM F.E.          Current
Id    Stmt type     Database           Lvl Mode       ERR  ERR  Vers Explain    Role
22     -            test               CR  Not Wait    0    0   9.03 Off         hr
Last parsed SQL statement :
  select id, name, decrypt_char(ssn, 'XXXXXXXXXX') from emp
```

Figure 14-62. onstat -g sql Output

You can interpret the output from this option as follows:

Sess id The session identifier

SQL Stmt type The type of SQL statement

Current Database
Name of the current database of the session

ISO Lvl Isolation level

- DR Dirty Read
- CR Committed Read
- CS Cursor Stability
- DRU Dirty Read, Retain Update Locks
- CRU Committed Read, Retain Update Locks
- CSU Cursor Stability, Retain Update Locks
- RR Repeatable Read
- NL Database Without Transactions

Lock mode Lock mode of the current session

SQL Error SQL error number encountered by the current statement

ISAM Error ISAM error number encountered by the current statement

F.E. Version Version of the client program

Explain SET EXPLAIN setting

Current Role Role of the current user

The onstat -g ssc Option

Monitors the number of times that the database server reads the SQL statement in the cache.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 00:08:26 -- 29696 Kbytes

Statement Cache Summary:
#lrus   currsize  maxsize   Poolsize  #hits   nolimit
4       117640    524288    139264    0       1
Statement Cache Entries:
lru hash ref_cnt hits flag heap_ptr     database           user
-----------------------------------------------------------------
  0  262       0    7   -F  aad8038     sscsi007           admin
   INSERT INTO ssc1 ( t1_char , t1_short , t1_key , t1_float , t1_smallfloat
    , t1_decimal , t1_serial ) VALUES ( ? , ? , ? , ? , ? , ? , ? )
  0  127       0    9   -F  b321438     sscsi007           admin
   INSERT INTO ssc2 ( t2_char , t2_key , t2_short ) VALUES ( ? , ? , ? )
  1  134       0   15   -F  aae0c38     sscsi007           admin
   SELECT t1_char , t1_short , t1_key , t1_float , t1_smallfloat ,
    t1_decimal , t1_serial FROM ssc1 WHERE t1_key = ?
  1  143       0    3   -F  b322c38     sscsi007           admin
   INSERT INTO ssc1 ( t1_char , t1_key , t1_short ) SELECT t2_char , t2_key
    + ? , t2_short FROM ssc2
  2   93       0    7   -F  aae9838     sscsi007           admin
   DELETE FROM ssc1 WHERE t1_key = ?
  2  276       0    7   -F  aaefc38     sscsi007           admin
   SELECT count ( * ) FROM ssc1
  2  240       1    7   -F  b332838     sscsi007           admin
   SELECT COUNT ( * ) FROM ssc1 WHERE t1_char = ? AND t1_key = ? AND
    t1_short = ?
  3   31       0    7   -F  aaec038     sscsi007           admin
   SELECT count ( * ) FROM ssc1 WHERE t1_key = ?
  3   45       0    1   -F  b31e438     sscsi007           admin
   DELETE FROM ssc1
  3  116       0    0   -F  b362038     sscsi007           admin
   SELECT COUNT ( * ) FROM ssc1
    Total number of entries: 10.
```

Figure 14-63. onstat -g ssc Output

Output Description:

Statement Cache Summary section

#lrus
: Number of least recently used queues (LRUS)

currsize
: Current cache size

maxsize
: Limit on total cache memory

Poolsize
: Total pool size

#hits
: The number of hits before insertion. This number equals the value of the STMT_CACHE_HITS configuration parameter

nolimit
: The value of the STMT_CACHE_NOLIMIT configuration parameter

The Statement Cache Entries section shows the entries that are fully inserted into the cache.

lru
: The index of lru queue to which the cache entry belongs

hash
: Hash values of cached entry

ref_count
: Number of threads referencing the statement

hits
: Number of times a statement matches a statement in the cache. The match can be for a key-only or fully cached entry.

flag Cache entry flag
-F indicates the statement is fully cached
-D indicates the statement is dropped

heap_ptr Address of memory heap for cache entry

The onstat -g stk tid Option

The **onstat -g stk tid** option dumps stack of thread specified by thread ID.

Example Output:

```
Stack for thread: 2 adminthd
 base: 0x000000010aad5028
  len:   33280
   pc: 0x00000001002821e8
  tos: 0x000000010aadc621
state: running
   vp: 2

0x1002821e8 oninit :: yield_processor + 0x260 sp=0x10aadce20(0x10ac834d0, 0x0, 0x1,
   0x100000000, 0xc8a000, 0x100c8a000)
0x100274e38 oninit :: wake_periodic + 0xdc sp=0x10aadced0 delta_sp=176(0x41b0, 0xc7a024bc,
   0x0, 0x41c4, 0x10aacf598, 0x90)
0x100274fcc oninit :: admin_thread + 0x108 sp=0x10aadcf80 delta_sp=176(0x0, 0x2328,
   0xd26c00, 0x5, 0xc8a000, 0x156c)
0x1002484ec oninit :: startup + 0xd8 sp=0x10aadd050 delta_sp=208(0xa, 0x10aad47d0,
   0x10aad47d0, 0x100db1988, 0xd1dc00, 0x1)
```

Figure 14-64. onstat -g stk Output

The onstat -g stm Option

Displays the memory that each prepared SQL statement uses. To display the memory for only one session, specify the session ID in the **onstat -g stm** option.

Example Output:

```
onstat -g stm 65
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 00:26:46 -- 29696 Kbytes
session   65 ------------------------------------------------------------
 sdblock  heapsz  statement ('*' = Open cursor)
 aad8028   16544   SELECT COUNT ( * ) FROM ssc1 WHERE t1_char = ?
 AND t1_key = ? AND t1_short = ?
```

Figure 14-65. onstat -g stm Output

Output Description:

sdblock Address of the statement descriptor block

heapsz Size of the statement memory heap

statement Query text

The onstat -g sts Option

The **onstat -g sts** option prints maximum and current stack use per thread.

Example Output:

```
Stack usage:

TID     Total   Max             Current         Thread Name
                bytes   %       bytes   %
2       32768   3124    9       3079    9       adminthd
3       32768   2870    8       2871    8       childthd
5       32768   14871   45      2871    8       Cosvr Avail Mgr
6       32768   2870    8       2871    8       dfm_svc
7       131072  3190    2       3191    2       xmf_svc
9       32768   3126    9       3127    9       xtm_svcc
10      32768   3580    10      3335    10      xtm_svcp
11      32768   3238    9       3239    9       cfgmgr_svc
12      32768   6484    19      2871    8       lio vp 0
14      32768   6484    19      2871    8       pio vp 0
16      32768   6484    19      2871    8       aio vp 0
18      131072  10391   7       2871    2       msc vp 0
20      32768   4964    15      2871    8       fifo vp 0
22      32768   4964    15      2871    8       fifo vp 1
24      32768   6028    18      2871    8       aio vp 1
26      32768   5444    16      2951    9       dfmxpl_svc
27      32768   2886    8       2887    8       sch_svc
28      32768   7812    23      5015    15      rqm_svc
29      32768   7140    21      3079    9       sm_poll
30      32768   11828   36      6439    19      sm_listen
31      32768   2870    8       2871    8       sm_discon
32      32768   14487   44      4055    12      main_loop()
33      32768   4272    13      2903    8       flush_sub(0)
34      32768   2902    8       2903    8       flush_sub(1)
35      32768   2870    8       2871    8       btscanner 0
36      32768   3238    9       3239    9       aslogflush
37      32768   3055    9       2887    8       bum_local
38      32768   3238    9       3239    9       bum_rcv
39      32768   4902    14      4903    14      onmode_mon
42      32768   4964    15      2871    8       lio vp 1
44      32768   5136    15      2871    8       pio vp 1
```

Figure 14-66. onstat -g sts Output

The onstat -g sync Option

Displays the synchronization status when Enterprise Replication is used. The -g sync option is used primarily as a debugging tool and by IBM Support.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.U      -- On-Line -- Up 00:10:16 -- 44084 Kbytes
Prim   Sync   St.  Shadow Flag Stat Block  EndBlk
Repl   Source      Repl             Num    Num

655361 20     0    1310729 2    0    592    600
```

Figure 14-67. onstat -g sync Output

Output Description:

Prim Repl
: Replicate number of the replicate being synchronized

Sync Source
: Source server of the sync

4 *St* Sync replicate state

4 *Shadow Repl*
4 The shadow replicate used to perform the sync

4 *Flag* Internal flags:
4 - 0x02 = external sync
4 - 0x04 = shutdown request has been issued
4 - 0x08 = abort has occured
4 - 0x010 = a replicate stop has been requested
4 - 0X020 = shadow or primary replicate has been deleted

4 *Stat* Resync job state

4 *Block num*
4 Last block applied on targets (on source always 0)

4 *EndBlock Num*
4 Last block in resync process. Marks the end of the sync scan on the target.
4 A value of' -2 indicates that the scan is still in progress, and the highest
4 block number is not yet known.

4 Additional fields for forwarded rows:

4 *ServID* Server where forwarded row originated

4 *fwdLog ID*
4 Originator's log ID of the forwarded row

4 *fwdLog POS*
4 Originator's log position of the forwarded row

4 *endLog ID*
4 Operation switches back to normal at this point

4 *endLog POS*
4 Operation switches back to normal at this log position

4 *complete flag*
4 Set to 1 after normal processing resumes for the originating source.

The onstat -g tpf tid Option

Prints thread profile for *tid*; 0 prints profiles for all threads.

Example Output:

```
onstat -g tpf 945
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 00:21:39 -- 29696 Kbytes
Thread profiles
tid lkreqs lkw dl to lgrs isrd iswr isrw isdl isct isrb lx bfr bfw lsus lsmx seq
945 1969   0   0  0  6181 1782 2069 13   0    0    0    0  16183 7348 743580 0    6
```

Figure 14-68. onstat -g tpf Output

Output Description:

tid Thread ID

lkreqs Lock requests

lkw Lock waits

dl Deadlocks

to Remote deadlock timeout

lgrs Log records

isrd Number of reads

iswr Number of writes

isrw Number of rewrites

isdl Number of deletes

isct Number of commits

isrb Number of rollbacks

lx Long transactions

bfr Buffer reads

bfw Buffer writes

lsus Log space currently used

lsmx Max log space used

seq Sequence scans

The onstat -g wmx Option

The **onstat -g wmx** option prints all mutexes with waiters.

Example Output:

```
Mutexes with waiters:
mid      addr             name              holder   lkcnt  waiter   waittime
```

Figure 14-69. onstat -g wmx Output

onstat -G

Use the **-G** option to display information about global transactions generated through TP/XA. For more information on TP/XA, see the *IBM Informix TP/XA Programmer's Manual*.

Example Output:

Figure 14-70 on page 14-78 shows an example of **onstat -G** output:

```
Global Transaction Identifiers
address  flags  isol   timeout fID     gtl  bql  data
c736618  --R-G  DIRTY   45   4478019 16   48   438709F23076254C80F33A62BAF4CF763
C1BCFFAD7AE0243AA5CE243FA5381C903AA9F52A1546044992C5A7BC03582E77999EFBA725D3D40B
DAF37404D9DAFF8
**br**   -----  DIRTY   45   4478019 16   48   438709F23076254C80F33A62BAF4CF763
C1BCFFAD7AE0243AA5CE243FA5381C903AA9F52A1546044992C5A7BC03582E77999EFBA725D3D40B
DAF37404D9DAFF8
 1 active, 128 total
```

Figure 14-70. onstat -G Output

Output Description:

The flag codes for position 1 (current transaction state):

A User thread attached to the transaction

S TP/XA suspended transaction

C TP/XA waiting for rollback

The flag codes for position 2 (transaction mode):

T Tightly-coupled mode (MTS)

L Loosely-coupled mode (default mode)

The flag codes for position 3 (transaction stage):

B Begin work

P Distributed query prepared for commit

X TP/XA prepared for commit

C Committing or committed

R Rolling back or rolled back

H Heuristically rolling back or rolled back

The flag code for position 4:

X XA DataSource global transaction

The flag codes for position 5 (type of transaction):

G Global transaction

C Distributed query coordinator

S Distributed query subordinate

B Both distributed query coordinator and subordinate

onstat -i

Use the **-i** option to put **onstat** in interactive mode. In interactive mode, you can enter multiple **onstat** options per session, but only one at a time. An **onstat** prompt appears and allows you to enter an option.

In interactive mode, do not precede the option with a dash.

Two additional options, **r** ***seconds*** and **rz** ***seconds***, are available in interactive mode. The **r** ***seconds*** option is similar to the current **onstat -r** ***seconds*** option, which repeatedly generates a display. If an administrator executes **r** ***seconds*** at the interactive-mode prompt, the prompt changes to reflect the specified interval in seconds and reappears, waiting for the next command. In the following example, the display generated by the next command repeats every three seconds:

```
onstat> r 3
onstat[3]>
```

The **rz** ***seconds*** option enables you to repeat the next command as specified and set all profile counters to 0 between each execution.

To terminate interactive mode, press `CTRL-d`.

To terminate a repeating sequence, press `CTRL-c`.

onstat -j

The **-j** option of the **onstat** utility provides special information about the status of an **onpload** job. The **-j** option provides an interactive mode that is analogous to **onstat -i**.

When **onpload** starts, it writes a series of messages to **stdout** or to a log file. The following lines show a typical **onpload** log file:

```
Mon Jul 24 16:11:30 1995

SHMBASE        0x4400000
CLIENTNUM      0x49010000
Session ID 1

Load Database   -> cnv001
Load Table      -> cnv001a
Load File       -> testrec.dat
Record Mapping  -> cnv001a

Database Load Completed -- Processed 50 Records
Records Inserted-> 50
Detected Errors--> 0
Engine Rejected--> 0

Mon Jul 24 16:11:37 1995
```

The two lines that start with SHMBASE and CLIENTNUM provide the information that you need to locate shared memory for an instance of **onpload**. The **oninit** process has similar values stored in the **$ONCONFIG** file. When you use **onstat** to gather information about the **oninit** process, **onstat** uses information from **$INFORMIXDIR/etc/$ONCONFIG** to locate shared memory. When you use **onstat** to gather information about **onpload**, you must give **onstat** the name of a file that contains SHMBASE and CLIENTNUM information.

Typically the file that contains the SHMBASE and CLIENTNUM information is the log file. For example, if the **onpload** log file is **/tmp/cnv001a.log**, you can enter the following command:

```
onstat -j /tmp/cnv001a.log
```

The previous command causes **onstat** to attach to **onpload** shared memory and to enter interactive mode. You can then enter ? or any other pseudo request to see a usage message displayed. An example follows:

```
onstat> ?
Interactive Mode: One command per line, and - are optional.
    -rz    repeat option every n seconds (default: 5) and
            zero profile counts
  MT COMMANDS:
    all   Print all MT information
    ath   Print all threads
    wai   Print waiting threads
    act   Print active threads
    rea   Print ready threads
    sle   Print all sleeping threads
    spi   print spin locks with long spins
    sch   print VP scheduler statistics
    lmx   Print all locked mutexes
    wmx   Print all mutexes with waiters
    con   Print conditions with waiters
    stk <tid>   Dump the stack of a specified thread
    glo   Print MT global information
    mem <pool name|session id>   print pool statistics.
    seg   Print memory segment statistics.
    rbm   print block map for resident segment
    nbm   print block map for non-resident segments
    afr <pool name|session id> Print allocated poolfragments.
    ffr <pool name|session id> Print free pool fragments.
    ufr <pool name|session id> Print pool usage breakdown
    iov   Print disk IO statistics by vp
    iof   Print disk IO statistics by chunk/file
    ioq   Print disk IO statistics by queue
    iog   Print AIO global information
    iob   Print big buffer usage by IO VP class
    sts   Print max and current stack sizes
    qst   print queue statistics
    wst   print thread wait statistics
    jal   Print all Pload information
    jct   Print Pload control table
    jpa   Print Pload program arguments
    jta   Print Pload thread array
    jmq   Print Pload message queues, jms for summary only
onstat>
```

Most of the options are the same as those that you use to gather information about Dynamic Server, with the following exceptions:

```
jal   Print all Pload information
jct   Print Pload control table
jpa   Print Pload program arguments
jta   Print Pload thread array
jmq   Print Pload message queues, jms for summary only
```

These options apply only to **onpload**. You can use **onstat -j** to check the status of a thread, locate the VP and its PID, and then attach a debugger to a particular thread. The options for **onstat** that do not apply to **onpload** are not available (for example, `-g ses`).

onstat -k

Use the **-k** option to display information about active locks.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1  -- On-Line -- Up 03:55:17 -- 15360 Kbytes

Locks
address  wtlist   owner    lklist   type     tblsnum  rowid    key#/bsiz
a095f78  0        a4d9e68  0        HDR+S    100002   203         0
 1 active, 2000 total, 2048 hash buckets, 0 lock table overflows
```

Figure 14-71. onstat -k Output

Output Description:

You can interpret output from this option as follows:

address
: Is the address of the lock in the lock table

  If a user thread is waiting for this lock, the address of the lock appears in the **wait** field of the **onstat -u** (users) output.

wtlist
: Is the first entry in the list of user threads that is waiting for the lock, if there is one

owner
: Is the shared-memory address of the thread that is holding the lock

  This address corresponds to the address in the **address** field of **onstat -u** (users) output.

lklist
: Is the next lock in a linked list of locks held by the owner just listed

type
: Uses the following codes to indicate the type of lock:

HDR	Header
B	Bytes
S	Shared
X	Exclusive
I	Intent
U	Update
IX	Intent-exclusive
IS	Intent-shared
SIX	Shared, intent-exclusive

tblsnum
: Is the tblspace number of the locked resource. If the number is less than 10000, it indicates Enterprise Replication pseudo locks.

rowid
: Is the row identification number

  The rowid provides the following lock information:
  - If the rowid equals zero, the lock is a table lock.
  - If the rowid ends in two zeros, the lock is a page lock.
  - If the rowid is six digits or fewer and does not end in zero, the lock is probably a row lock.
  - If the rowid is more than six digits, the lock is probably an index key-value lock.

key#/bsiz
: Is the index key number, or the number of bytes locked for a VARCHAR lock

If this field contains 'K-' followed by a value, it is a key lock. The value identifies which index is being locked. For example, K-1 indicates a lock on the first index defined for the table.

The maximum number of locks available is specified as LOCKS in the **ONCONFIG** file.

onstat -l

Use the **-l** option to display information about physical and logical logs.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 03:55:32 -- 15360 Kbytes

Physical Logging
Buffer bufused  bufsize  numpages numwrits pages/io
  P-1  0        16       716      55       13.02
      phybegin         physize  phypos   phyused  %used
      1:263            500      270      0        0.00

Logical Logging
Buffer bufused  bufsize  numrecs  numpages numwrits recs/pages pages/io
  L-3  0        16       42169    2872     1043     14.7       2.8
        Subsystem    numrecs  Log Space used
        OLDRSAM      42169    4436496

address  number   flags    uniqid   begin                size     used    %used
a517f70  1        U-B----  1        1:763                 500      500   100.00
a517fb0  2        U-B----  2        1:1263                500      500   100.00
a40daf0  3        U-B----  3        1:1763                500      500   100.00
a40db30  4        U-B----  4        1:2263                500      500   100.00
a40db70  5        U-B----  5        1:2763                500      500   100.00
a40dbb0  6        U---C-L  6        1:3263                500      372    74.40
a40dbf0  7        A------  0        1:3763                500        0     0.00
a40dc30  8        A------  0        1:4263                500        0     0.00
 8 active, 8 total
```

Figure 14-72. onstat -l Output

Output Description:

You can interpret output from this option as follows. The first section of the display describes the physical-log configuration:

buffer Is the number of the physical-log buffer

bufused Is the number of pages of the physical-log buffer that are used

bufsize Is the size of each physical-log buffer in pages

numpages Is the number of pages written to the physical log

numwrits Is the number of writes to disk

pages/io Is calculated as `numpages/numwrits`

This value indicates how effectively physical-log writes are being buffered.

phybegin Is the physical page number of the beginning of the log

physize Is the size of the physical log in pages *phypos* Is the current position in the log where the next log-record write is to occur

phyused Is the number of pages used in the log

%used Is the percent of pages used

The second section of the **onstat -l** display describes the logical-log configuration:

buffer Is the number of the logical-log buffer

bufused Is the number of pages used in the logical-log buffer

bufsize Is the size of each logical-log buffer in pages

numrecs Is the number of records written

numpages Is the number of pages written

numwrits Is the number of writes to the logical log

recs/pages Is calculated as `numrecs/numpages`

You cannot affect this value. Different types of operations generate different types (and sizes) of records.

pages/io is calculated as `numpages/numwrits`

You can affect this value by changing the size of the logical-log buffer (specified as LOGBUFF in the ONCONFIG file) or by changing the logging mode of the database (from buffered to unbuffered, or vice versa).

The following fields are repeated for each logical-log file:

address Is the address of the log-file descriptor

number Is logid number for the logical-log file

The logid numbers might be out of sequence because either the database server or administrator can insert a log file in-line.

flags Provides the status of each log as follows:

- A Newly added (and ready to use)
- B Backed up
- C Current logical-log file
- D Marked for deletion

  To drop the log file and free its space for reuse, you must perform a level-0 backup of all storage spaces
- F Free, available for use
- L The most recent checkpoint record
- U Used

uniqid Is the unique ID number of the log

begin Is the beginning page of the log file

size Is the size of the log in pages

used Is the number of pages used

%used Is the percent of pages used

active Is the number of active logical logs

total Is the total number of logical logs

The database server uses *temporary logical logs* during a warm restore because the permanent logs are not available then. The following fields are repeated for each temporary logical-log file:

address Is the address of the log-file descriptor

number Is logid number for the logical-log file

flags Provides the status of each log as follows:

- B Backed up
- C Current logical-log file
- F Free, available for use
- U Used

uniqid Is the unique ID number of the log

begin Is the beginning page of the log file

size Is the size of the log in pages

used Is the number of pages used

%used Is the percent of pages used

active Is the number of active temporary logical logs

onstat -m

Use the **-m** option to display the 20 most-recent lines of the system message log. You can use the **onstat -m** option with the database server in any mode, including offline.

Output from this option lists the full pathname of the message-log file and the 20 file entries. A date-and-time header separates the entries for each day. A time stamp prefaces single entries within each day. The name of the message log is specified as MSGPATH in the **ONCONFIG** file.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1  -- On-Line -- Up 03:55:41 -- 15360 Kbytes

Message Log File: /work/9.40/dbspaces/star3.log
14:41:00  Fuzzy Checkpoint Completed:  duration was 0 seconds, 1 buffers not flushed,
timestamp: 56447.
14:41:00  Checkpoint loguniq 6, logpos 0x17004c, timestamp: 56447

14:41:00  Maximum server connections 1
15:36:00  Fuzzy Checkpoint Completed:  duration was 0 seconds, 1 buffers not flushed,
timestamp: 56477.
15:36:00  Checkpoint loguniq 6, logpos 0x17104c, timestamp: 56477

15:36:00  Maximum server connections 1
16:31:00  Fuzzy Checkpoint Completed:  duration was 0 seconds, 1 buffers not flushed,
timestamp: 56512.
16:31:00  Checkpoint loguniq 6, logpos 0x17204c, timestamp: 56512

16:31:00  Maximum server connections 1
17:26:00  Fuzzy Checkpoint Completed:  duration was 0 seconds, 1 buffers not flushed,
timestamp: 56542.
17:26:00  Checkpoint loguniq 6, logpos 0x17304c, timestamp: 56542

17:26:00  Maximum server connections 1
```

Figure 14-73. onstat -m Output

onstat -O

Use the -**O** option of the **onstat** utility to display information about the Optical Subsystem memory cache and staging-area blobspace. You can interpret output from this option as follows. The totals shown in the display accumulate from session to session. The database server resets the totals to `0` only when you execute **onstat -z**.

Example Output:

```
IBM Informix Dynamic Server Version 9.40.UC1 --Online-- Up 00:45:18 -- 11656 Kbytes

Optical StageBlob Cache
System Cache Totals:                             System Blob Totals:
Size    Alloc.  Avail.           Number   Kbytes  Number  Kbytes
500     500     0                1        20      3       1500

User Cache Totals:                               User Blob Totals:
SID     User    Size             Number   Kbytes  Number  Kbytes
94      doug    250              1        20      1       300
95      beth    500              0        0       2       1200
```

Figure 14-74. onstat -O Output

Output Description:

The first section of the display provides the following information on system-cache totals:

size Is the size that the OPCACHEMAX configuration parameter specifies *alloc* Is the number of 1-kilobyte allocations to the cache

avail Describes how much of **alloc** (in kilobytes) is not used

number Is the number of simple large objects that the database server successfully put in the cache without overflowing

kbytes Is the number of kilobytes of TEXT or BYTE data that the database server put in the cache without overflowing

number Is the number of simple large objects that the database server wrote to the staging-area blobspace

kbytes Is the number of kilobytes of TEXT or BYTE data that the database server wrote to the staging-area blobspace

Although the **size** output indicates the amount of memory that is specified in the configuration parameter OPCACHEMAX, the database server does not allocate memory to OPCACHEMAX until necessary. Therefore, the **alloc** output reflects only the number of 1-kilobyte allocations of the largest simple large object that has been processed. When the values in the **alloc** and **avail** output are equal to each other, the cache is empty.

The second section of the display describes the following user-cache totals information:

SID Is the session ID for the user

user Is the user ID of the client

size Is the size specified in the **INFORMIXOPCACHE** environment variable, if it is set

If you do not set the **INFORMIXOPCACHE** environment variable, the database server uses the size that you specify in the configuration parameter OPCACHEMAX.

number Is the number of simple large objects that the database server put into cache without overflowing

kbytes Is the number of kilobytes of TEXT or BYTE data that the database server put in the cache without overflowing

number Is the number of simple large objects that the database server wrote to the staging-area blobspace

kbytes Is the number of kilobytes of TEXT or BYTE data that the database server wrote to the staging-area blobspace

The last line of the display lists the total number of sessions that are using the cache.

onstat -p

Use the **-p** option to display profile counts either since you started the database server or since you ran **onstat** with the **-z** option.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1  -- On-Line -- Up 03:56:40 -- 15360 Kbytes

Profile
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
939      943      143905   99.35   3925     10816    46919    91.63

isamtot  open     start    read     write    rewrite  delete   commit   rollbk
100055   15851    16112    24632    13343    1342     1392     905      0

gp_read  gp_write gp_rewrt gp_del   gp_alloc gp_free  gp_curs
0        0        0        0        0        0        0

ovlock   ovuserthread ovbuff   usercpu  syscpu   numckpts flushes
0        0            0        12.00    2.69     9        101

bufwaits lokwaits lockreqs deadlks  dltouts  ckpwaits compress seqscans
8        0        26894    0        0        1        1247     478

ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
5        0        10       15         23
```

Figure 14-75. onstat -p Output

Output Description:

The first portion of the display describes reads and writes.

Reads and writes are tabulated in three categories: from disk, from buffers, and number of pages (read or written).

The first **%cached** field is a measure of the number of reads from buffers compared to reads from disk. The second **%cached** field is a measure of the number of writes to buffers compared to writes to disk.

The database server buffers information and writes to the disk in pages. For this reason, the number of disk writes displayed as `dskwrits` is usually less than the number of writes that an individual user executes:

dskreads Is the number of actual reads from disk

pagreads Is the number of pages read

bufreads Is the number of reads from shared memory

%cached Is the percent of reads cached, calculated as follows:

```
100 * (bufreads - dskreads) / bufreads
```

If `bufreads` exceeds the maximum integer (or long) value, its internal representation becomes a negative number, but the value appears as `0.0`.

dskwrits Is the actual number of physical writes to disk

This number includes the writes for the physical and logical logs reported in **onstat -l**.

pagwrits Is the number of pages written

bufwrits Is the number of writes to shared memory

%cached Is the percent of writes cached, calculated as follows:

```
100 *(bufwrits - dskwrits) / bufwrits
```

If `dskwrits` exceeds `bufwrits`, the value appears as `0.0`. The next portion of the **-p** display tabulates the number of times different ISAM calls were executed. The calls occur at the lowest level of operation and do not necessarily correspond one-to-one with SQL statement execution. A single query might generate multiple ISAM calls. These statistics are gathered across the database server and cannot be used to monitor activity on a single database unless only one database is active or only one database exists:

isamtot
: Is the total number of calls

open
: Increments when a tblspace is opened

start
: Increments the pointer within an index

read
: Increments when the read function is called

write
: Increments with each write call

rewrite
: Increments when an update occurs

delete
: Increments when a row is deleted

commit
: Increments each time that an **iscommit()** call is made

No one-to-one correspondence exists between this value and the number of explicit COMMIT WORK statements that are executed.

rollbk
: Increments when a transaction is rolled back

The next portion of the **-p** display provides information on generic pages. The Generic Page Manager provides an API for Dynamic Server to manage nonstandard pages in the database server buffer pool. The following table describes the Generic Page Manager fields in the **onstat -p** output.

gp_read
: The number of generic page reads

gp_write
: The number of generic page writes

gp_rewrt
: The number of generic page updates

gp_del
: The number of generic page deletes

gp_alloc
: The number of generic page allocations

gp_free
: The number of generic pages freed and returned to tblspaces

gp_curs
: The number of cursors used against generic pages

The next portion of the **-p** display tracks the number of times that a resource was requested when none was available:

ovlock
: Is the number of times that the database server attempted to allocate locks more than 15 times

For more information, see "LOCKS" on page 1-56.

ovuserthread
: Is the number of times that a user attempted to exceed the maximum number of user threads

ovbuff
: Is the number of times that the database server could not find a free shared-memory buffer

When no buffers are free, the database server writes a dirty buffer to disk and then tries to find a free buffer.

usercpu Is the total user CPU time that all user threads use, expressed in seconds

This entry is updated every 15 seconds.

syscpu Is the total system CPU time that all user threads use, expressed in seconds

This entry is updated every 15 seconds.

numckpts Is the number of checkpoints since the boot time

flushes Is the number of times that the buffer pool has been flushed to the disk

The next portion of the **-p** display contains miscellaneous information, as follows:

bufwaits Increments each time that a user thread must wait for a buffer

lokwaits Increments each time that a user thread must wait for a lock

lockreqs Increments each time that a lock is requested

deadlks Increments each time that a potential deadlock is detected and prevented

dltouts Increments each time that the distributed deadlock time-out value is exceeded while a user thread is waiting for a lock

ckpwaits Is the number of checkpoint waits

compress Increments each time that a data page is compressed

seqscans Increments for each sequential scan

The last portion of the **-p** display contains the following information:

ixda-RA Is the count of read-aheads that go from index leaves to data pages

idx-RA Is the count of read-aheads that traverse index leaves

da-RA Is the count of data-path-only scans

RA-pgsused Indicates the number of pages used that the database server read ahead

If this number is significantly less than the total number of pages read ahead, the read-ahead parameters might be set too high.

lchwaits Stores the number of times that a thread was required to wait for a shared-memory latch

A large number of latch waits typically results from a high volume of processing activity in which the database server is logging most of the transactions.

onstat -P

Use the **-P** option to display for all partitions the partition number and the pages in the buffer pool that belong to the partition.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 18:44:15 -- 34816 Kbytes
Buffer pool page size: 2048
partnum  total    btree    data     other    dirty
0        36       1        8        27       0
1048577  2        0        0        2        0
1048578  4        1        1        2        0
1048579  23       10       12       1        0
1048580  68       31       36       1        0
4194309  3        0        1        2        0

Totals:  3000     786      1779     435      0
Percentages:
Data  59.30
Btree 26.20
Other 14.50

Buffer pool page size: 8192
partnum  total    btree    data     other    dirty
0        999      0        0        999      0
5242881  1        0        0        1        0

Totals:  1000     0        0        1000     0
Percentages:
Data  0.00
Btree 0.00
Other 100.00
```

Figure 14-76. onstat -P Output

Output Description:

Buffer pool page size
Is the size of the buffer pool pages in bytes

partnum Is the partition number

total Is the total number of partitions

btree Is the number of B-tree pages in the partition

data Is the number of data pages in the partition

other Is the number of other pages in the partition

resident Is the number of resident pages in the partition

dirty Is the number of dirty pages in the partition

onstat -R

Use the **-R** option to display detailed information about the LRU queues, FLRU queues, and MLRU queues. For an in-depth discussion of the three types of queues, see LRU queues in the shared-memory chapter of the *IBM Informix Administrator's Guide*.

For each queue, **onstat -R** lists the number of buffers in the queue and the number and percentage of buffers that have been modified.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 18:46:59 -- 34816 Kbytes
    Buffer pool page size: 2048
    8 buffer LRU queue pairs              priority levels
# f/m   pair total     % of    length       LOW      HIGH
 0 f        375     100.0%      375        375         0
 1 m                  0.0%        0          0         0
 2 f        375     100.0%      375        375         0
 3 m                  0.0%        0          0         0
 4 f        375     100.0%      375        375         0
 5 m                  0.0%        0          0         0
 6 F        375     100.0%      375        375         0
 7 m                  0.0%        0          0         0
 8 f        375     100.0%      375        375         0
 9 m                  0.0%        0          0         0
10 f        375     100.0%      375        375         0
11 m                  0.0%        0          0         0
12 f        375     100.0%      375        375         0
13 m                  0.0%        0          0         0
14 f        375     100.0%      375        375         0
15 m                  0.0%        0          0         0
0 dirty, 3000 queued, 3000 total, 4096 hash buckets, 2048 buffer size
start clean at  60.000% (of pair total) dirty, or 226 buffs dirty, stop at
  50.000%
Buffer pool page size: 8192
    4 buffer LRU queue pairs              priority levels
# f/m   pair total     % of    length       LOW      HIGH
 0 F        250     100.0%      250        250         0
 1 m                  0.0%        0          0         0
 2 f        250     100.0%      250        250         0
 3 m                  0.0%        0          0         0
 4 f        250     100.0%      250        250         0
 5 m                  0.0%        0          0         0
 6 f        250     100.0%      250        250         0
 7 m                  0.0%        0          0         0
0 dirty, 1000 queued, 1000 total, 1024 hash buckets, 8192 buffer size
start clean at  60.000% (of pair total) dirty, or 150 buffs dirty, stop at
  50.000%
```

Figure 14-77. onstat -R Output

Output Description:

You can interpret output from this option as follows:

Buffer pool page size
: Is the page size of the buffer pool in bytes

#
: Shows the queue number

  Each LRU queue is composed of two subqueues: an FLRU queue and a MLRU queue. (For a definition of FLRU and MLRU queues, see LRU queues in the shared-memory chapter of the *IBM Informix Administrator's Guide*.) Thus, queues 0 and 1 belong to the first LRU queue, queues 2 and 3 belong to the second LRU queue, and so on.

f/m
: Identifies queue type

  This field has four possible values:

  f
  : Free LRU queue

    In this context, free means not modified. Although nearly all the buffers in an LRU queue are available for use, the database server attempts to use buffers from the FLRU queue rather than the MLRU queue. (A modified buffer must be written to disk before the database server can use the buffer.)

F
: Free LRU with fewest elements

  The database server uses this estimate to determine where to put unmodified (free) buffers next.

m
: MLRU queue

M
: MLRU queue that a flusher is cleaning

length
: Tracks the length of the queue measured in buffers

% of
: Shows the percent of LRU queue that this subqueue composes

  For example, suppose that an LRU queue has 50 buffers, with 30 of those buffers in the MLRU queue and 20 in the FLRU queue. The **% of** column would list percents of 60.00 and 40.00, respectively.

pair total
: Provides the total number of buffers in this LRU queue

priority levels
: Displays the priority levels: `LOW`, `MED_LOW`, `MED_HIGH`, `HIGH`

The **-R** option also lists the priority levels.

Summary information follows the individual LRU queue information. You can interpret the summary information as follows:

dirty
: Is the total number of buffers that have been modified in all LRU queues

queued
: Is the total number of buffers in LRU queues

total
: Is the total number of buffers

hash buckets
: Is the number of hash buckets

buffer size
: Is the size of each buffer

start clean
: Is the value of LRU_MAX_DIRTY

stop at
: Is the value of LRU_MIN_DIRTY

priority downgrades
: Is the number of LRU queues downgraded to a lower priority.

priority upgrades
: Is the number of LRU queues upgraded to a higher priority.

onstat -s

Use the **-s** option to display general latch information.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 03:57:17 -- 15360 Kbytes

Latches with lock or userthread set
name     address  lock wait userthread
```

Figure 14-78. onstat -s Output

Output Description:

You can interpret output from this option as follows:

name
: Identifies the resource that the latch controls with the following abbreviations:

archive	Storage-space backup
bf	Buffers
bh	Hash buffers
chunks	Chunk table
ckpt	Checkpoints
dbspace	Dbspace table
flushctl	Page-flusher control
flushr	Page cleaners
locks	Lock table
loglog	Logical log
LRU	LRU queues
physb1	First physical-log buffer
physb2	Second physical-log buffer
physlog	Physical log
pt	Tblspace tblspace
tblsps	Tblspace table
users	User table

address
: Is the address of the latch

This address appears in the **-u** (users) output wait field if a thread is waiting for the latch.

lock
: Indicates if the latch is locked and set

The codes that indicate the lock status (`1` or `0`) are computer dependent.

wait
: Indicates if any user thread is waiting for the latch

userthread
: Is the shared-memory address of any user thread that is waiting for a latch

Instead this field contains the thread-control block address, which all threads have. You can compare this address with the user addresses in the **onstat -u** output to obtain the user-process identification number.

To obtain the **rstcb** address from the **tcb** address, examine the output of the **onstat -g ath** option, which lists both addresses for each user thread.

onstat -t and -T

Use the **-t** option to display tblspace information for active tblspaces, including whether tblspaces are memory resident. Use the **-T** option to display the total number of tblspaces.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 03:58:08 -- 15360 Kbytes

Tblspaces
 n address  flgs ucnt tblnum   physaddr         npages nused  npdata nrows  nextns
62 a40dc70  0    1    100001   1:14             250    250    0      0      1
195 ac843e0  0    1    1000df   1:236            16     9      4      53     2
 2 active, 221 total
```

Figure 14-79. onstat -t Output

Output Description:

You can interpret output from this option as follows:

n Is a counter of open tblspaces

address Is the address of the tblspace in the shared-memory tblspace table

flgs Uses the following flag bits to describe the flag:

0x00000001	Partition structure is being initialized
0x00000002	Partition was modified. The modified pages have not been flushed to disk.
0x00000004	Partition is being dropped
0x00000008	Partition is for a pseudo table
0x00000010	Partition is being altered in an ADD INDEX or DROP INDEX operation
0x00000020	Partition is being altered in an ALTER TABLE operation
0x00000080	Partition is being dropped while the dbspace is down
0x00000100	Simple large objects in blobspaces are not deleted when the table is dropped
0x00000200	Partition alter page count is updated
0x00000400	Pages have been altered to the latest database schema
0x00000800	System temp table
0x00001000	User temp table
0x00002000	Partition is resident
0x00004000	Index operations are deferred during recovery
0x00008000	Partition is being truncated
0x00010000	Partition is partially truncated

ucnt Is the usage count, which indicates the number of user threads currently accessing the tblspace

tblnum Is the tblspace number expressed as a hexadecimal value

The integer equivalent appears as the **partnum** value in the **systables** system catalog table.

physaddr Is the physical address (on disk) of the tblspace

npages Is the number of pages allocated to the tblspace

nused Is the number of used pages in the tblspace

npdata Is the number of data pages used

nrows Is the number of data rows used

nextns Is the number of noncontiguous extents allocated

This number is not the same as the number of times that a next extent has been allocated.

resident Indicates whether tblspace is memory-resident; 1 = yes, 0 = no

The **-t** option also lists the number of active tblspaces and the total number of tblspaces.

onstat -u

Use the **-u** option to print a profile of user activity.

Example Output:

```
Userthreads
address  flags   sessid   user     tty      wait     tout locks nreads   nwrites
a4d8018  ---P--D 1        informix -        0        0    0     58       4595
a4d8628  ---P--F 0        informix -        0        0    0     0        2734
a4d8c38  ---P--- 5        informix -        0        0    0     0        1
a4d9248  ---P--B 6        informix -        0        0    0     40       0
a4d9858  ---P--D 7        informix -        0        0    0     0        0
a4d9e68  Y--P--- 21       niraj    -        a65e5a8  0    1     0        0
 6 active, 128 total, 7 maximum concurrent
```

Figure 14-80. onstat -u Output

Output Description:

The **-u** option provides the following output for each user thread.

address Is the shared-memory address of the user thread (in the user table)

Compare this address with the addresses displayed in the **-s** output (latches); the **-b**, **-B**, and **-X** output (buffers); and the **-k** output (locks) to learn what resources this thread is holding or waiting for.

flags Provides the status of the session.

The flag codes for position 1:

B Waiting for a buffer

C Waiting for a checkpoint

G — Waiting for a write of the logical-log buffer

L — Waiting for a lock

S — Waiting for mutex

T — Waiting for a transaction

Y — Waiting for condition

X — Waiting for a transaction cleanup (rollback)

DEFUNCT — The thread has incurred a serious assertion failure, and has been suspended to allow other threads to continue their work.

The flag code for position 2:

* — Transaction active during an I/O failure

The flag code for position 3:

A — A dbspace backup thread

For other values that appear here, see the third position of flag codes for the **-x** option.

The flag code for position 4:

P — Primary thread for a session

The flag codes for position 5:

R — Reading

X — Thread in critical section

The flag codes for position 7:

B — A B-tree cleaner thread

C — Terminated user thread waiting for cleanup

D — A daemon thread

F — A page-cleaner thread

M — Special ON–Monitor thread (UNIX)

sessid — Is the session identification number

During operations such as parallel sorting and parallel index building, a session might have many user threads associated with it. For this reason, the session ID identifies each unique session.

user — Is the user login name (derived from the operating system)

tty — Indicates the tty that the user is using (derived from the operating system)

This field is blank on Windows.

wait — If the user thread is waiting for a specific latch, lock, mutex, or condition, this field displays the address of the resource. Use this address to map to information provided in the **-s** (latch) or **-k** (lock) output. If the wait is for a persistent condition, run a **grep** for the address in the **onstat -a** output.

tout Is the number of seconds left in the current wait

If the value is `0`, the user thread is not waiting for a latch or lock. If the value is `-1`, the user thread is in an indefinite wait.

locks Is the number of locks that the user thread is holding

(The **-k** output should include a listing for each lock held.)

nreads Is the number of disk reads that the user thread has executed

nwrites Is the number of write calls that the user thread has executed

All write calls are writes to the shared-memory buffer cache.

The last line of **onstat -u** output displays the maximum number of concurrent user threads that were allocated since you initialized the database server. For example, the last line of a sample **onstat -u** output is as follows:

```
4 active, 128 total, 17 maximum concurrent
```

The last part of the line, `17 maximum concurrent`, indicates that the maximum number of user threads that were running concurrently since you initialized the database server is 17.

The output also indicates the number of active users and the maximum number of users allowed.

onstat -x

Use the **-x** option to display transaction information on the database server. The transaction information is required only in the following situations:

- X/Open environment
- Database server participation in distributed queries
- Database server uses the Microsoft Transaction Server (MTS) transaction manager

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1  -- On-Line -- Up 03:58:41 -- 15360 Kbytes

Transactions
address  flags userthread locks  beginlg curlog  logposit   isol    retrys coord
a509018  A---- a4d8018    0      0       6       0x17304c   COMMIT  0
a5091e8  A---- a4d8628    0      0       0       0x0        COMMIT  0
a5093b8  A---- a4d8c38    0      0       0       0x0        COMMIT  0
a509588  A---- a4d9248    0      0       0       0x0        COMMIT  0
a509758  A---- a4d9858    0      0       0       0x0        COMMIT  0
a509928  A---S a4d9e68    1      0       0       0x0        COMMIT  0      xps_qa
6 active, 128 total, 8 maximum concurrent
```

Figure 14-81. onstat -x Output

Output Description:

You can interpret output from **onstat -x** as follows:

address Is the shared-memory address of the transaction structure

flags The flag codes for position 1 (current transaction state):

A User thread attached to the transaction

S TP/XA suspended transaction

C TP/XA waiting for rollback

The flag codes for position 2 (transaction mode):

T Tightly-coupled mode (MTS)

L Loosely-coupled mode (default mode)

The flag codes for position 3 (transaction stage):

B Begin work

P Distributed query prepared for commit

X TP/XA prepared for commit

C Committing or committed

R Rolling back or rolled back

H Heuristically rolling back or rolled back

The flag code for position 4:

X XA transaction

The flag codes for position 5 (type of transaction):

G Global transaction

C Distributed query coordinator

S Distributed query subordinate

B Both distributed query coordinator and subordinate

userthread Is the thread that owns the transaction (**rstcb** address)

locks Is the number of locks that the transaction holds

beginlg Is the log in which the BEGIN WORK record was logged

curlog Is the current log that the transaction is writing to

logposit Is the log position

The format of a 4-byte log position is `0xPPPPPBBB`, where `PPPPP` is the page offset in the log and `BBB` is the byte offset in the page. The *logposit* can refer to a maximum of 0x100000 (or 1048576) pages in a log file.

For example, a record on the first page of log `12`, at a byte offset of `24` would have a log position of `0x18` (page `0`, byte offset 18). For more information, see "Determining the Position of a Logical-Log Record" on page 14-99.

isol Is the isolation level.

retrys Are the attempts to start a recovery thread for the distributed query

coord Is the name of the transaction coordinator when the subordinate is executing the transaction

This field tells you which database server is coordinating the two-phase commit.

The last line of the **onstat -x** output indicates that `8` is the maximum number of concurrent transactions since you initialized the database server.

```
8 active, 128 total, 8 maximum concurrent
```

Determining the Position of a Logical-Log Record

The **curlog** and **logposit** fields provide the exact position of a logical-log record. If a transaction is not rolling back, **curlog** and **logposit** describe the position of the most recently written log record. When a transaction is rolling back, these fields describe the position of the most recently "undone" log record. As the transaction rolls back, the **curlog** and **logposit** values decrease. In a long transaction, the rate at which the **logposit** and **beginlg** values converge can help you estimate how much longer the rollback is going to take.

For an **onstat -x** example, see monitoring a global transaction in the chapter on multiphase commit protocols in the *IBM Informix Administrator's Guide*.

Determining the Mode of a Global Transaction

The **onstat -x** utility is useful for determining whether a global transaction is executing in loosely-coupled or tightly-coupled mode. The second position of the flags column displays the flags for global transactions. The `T` flag indicates tightly-coupled mode and the `L` flag indicates loosely-coupled mode.

Loosely-coupled mode means that the different database servers coordinate transactions but do not share locks. Each branch in a global transaction has a separate transaction XID. The records from all branches display as separate transactions in the logical log.

Tightly-coupled mode means that the different database servers coordinate transactions and share resources such as locking and logging. In a global transaction, all branches that access the same database share the same transaction XID. Log records for branches with the same XID appear under the same session ID. MTS uses tightly-coupled mode.

onstat -X

Use the **-X** option to obtain precise information about the threads that are waiting for buffers. For each buffer in use, the **-X** option displays general buffer information that is also available with either the **-b** or **-B** option. For more information, refer to **onstat -b** in "onstat -b" on page 14-8.

Example Output:

```
IBM Informix Dynamic Server Version 10.00.UC1 -- On-Line -- Up 18:47:42 -- 34816 Kbytes
    Buffers (Access)
address  owner    flags pagenum          memaddr  nslots pgflgs scount   waiter
    Buffer pool page size: 2048
 0 modified, 3000 total, 4096 hash buckets, 2048 buffer size
    Buffer pool page size: 8192
 0 modified, 1000 total, 1024 hash buckets, 8192 buffer size
```

Figure 14-82. onstat -X Output

Output Description:

The **onstat -X** option has a **waiter** field to list all user threads that are waiting for the buffer, whereas the **onstat -b** and **-B** options contain a **waitlist** field that displays the address of the first user thread that is waiting for the buffer. The maximum number of shared buffers is specified in the **buffers** field in the BUFFERPOOL configuration parameter in the ONCONFIG file.

Buffer pool page size
: is the size of the buffer pool pages in bytes

address
: Is the address of the buffer header in the buffer table

flags
: Flags identifying the current status of the buffer page:

0x01 Modified Data

0x02 Data

0x04 LRU

0x08 Error

0x20 LRU AIO write in progress

0x40 Chunk write in progress

0x80 Buffer is/will be result of read-ahead

0x100 Cleaner assigned to LRU

0x200 Buffer should avoid bf_check calls

0x400 Do log flush before writing page

0x800 Buffer has been 'buff' -checked

0x8000 Buffer has been pinned

0x10000
Buffer modified by fuzzy operation.

pagenum
: Is the physical page number on the disk

memaddr
: Is the buffer memory address

nslots
: Is the number of slot-table entries in the page

This field indicates the number of rows (or portions of a row) that are stored on the page.

pgflgs
: Uses the following values, alone or in combination, to describe the page type:

1	Data page
2	Tblspace page
4	Free-list page
8	Chunk free-list page

9	Remainder data page
b	Partition resident blobpage
c	Blobspace resident blobpage
d	Blob chunk free-list bit page
e	Blob chunk blob map page
10	B-tree node page
20	B-tree root-node page
40	B-tree branch-node page
80	B-tree leaf-node page
100	Logical-log page
200	Last page of logical log
400	Sync page of logical log
800	Physical log
1000	Reserved root page
2000	No physical log required
8000	B-tree leaf with default flags

scount Displays the number of threads that are waiting for the buffer

waiter Lists the addresses of all user threads that are waiting for the buffer

onstat -z

Use the **-z** option to clear database server statistics, including statistics that relate to Enterprise Replication, and set the profile counts to `0`.

If you use the **-z** option to reset and monitor the count of some fields, be aware that profile counts are incremented for all activity that occurs in any database that the database server manages. Any user can reset the profile counts and thus interfere with monitoring that another user is conducting.

Return Codes on Exit

The **onstat** utility returns the following codes on exit.

```
GLS failures: -1
Failed to attach shared memory: -1
Failed to attach shared memory when running 'onstat -': 255
All other errors detected by onstat: 1
No errors detected by onstat: 0
```

Chapter 15. The ontape Utility

ontape: Log, Back Up, and Restore

The **ontape** utility lets you perform the following tasks:

- Back up data that the database server manages
- Change database-logging status
- Back up logical-log files
- Start continuous logical-log file backups
- Restore data from a backup tape
- Use data replication

On UNIX, you must be logged in as user **root** or user **informix** to execute **ontape**. On Windows, you must be a member of the **Informix-Admin** group. For information about **ontape** and ON–Bar, see the *IBM Informix Backup and Restore Guide*.

Syntax

```
                                     (1)
>>--ontape--+-| Back Up Database Server |---------------------+-------><
            |                                    (2)          |
            +-| Change Database- Logging Status |-----------+
            |                             (3)                 |
            +-| Back Up Logical-Log Files |-------------------+
            |                                         (4)     |
            +-| Start Continuous Logical-Log Backups |--------+
            |                              (5)                |
            +-| Restore Data from Backup |--------------------+
            |                                           (6)   |
            '-| Performing an External Physical Restore |-----'
```

Notes:

1 See the *IBM Informix Backup and Restore Guide*

2 See the *IBM Informix Administrator's Guide*

3 See the *IBM Informix Backup and Restore Guide*

4 See the *IBM Informix Backup and Restore Guide*

5 See the *IBM Informix Backup and Restore Guide*

6 See the *IBM Informix Backup and Restore Guide*

Syntax for **ontape** other than the **-t** and the **-l** options appears in the *IBM Informix Administrator's Guide*.

Things to Consider

If more than one tape is needed during data replication, **ontape** prompts for each additional tape. Do not run **ontape** in background mode because you might need to provide input from the terminal or window.

Exit Codes

The **ontape** utility has two exit codes:

0	indicates a normal exit from **ontape.**
1	indicates an exceptional condition.

Part 3. Appendixes

Appendix A. Files That the Database Server Uses

This appendix provides brief summaries of the files that you use when you configure and use the database server. It also includes descriptions of files (and one directory) created and used internally by the database server. For many of these files, your only responsibilities are to recognize that those files are legitimate and refrain from deleting them.

Pathnames that appear in the following format indicate files that reside on UNIX: **/directory/filename**. Pathnames that appear in the following format indicate files that reside on Windows: **\directory\filename**.

In some cases, environment variables are used to specify the initial pathname of a file. On UNIX, references to environment variables begin with a dollar sign: **$INFORMIXDIR**. On Windows, references to environment variables begin and end with percent signs: **%INFORMIXDIR%**.

Database Server Files

Table A-1 lists the database server files and the directories in which they reside.

Table A-1. List of Files That the Database Server Uses

Filename	Directory	Purpose	Created
af.xxx	Specified by DUMPDIR configuration parameter	Assertion-failure information	By the database server
ac_msg.log	**/tmp**, **%INFORMIXDIR%\etc**	**archecker** message log (for Technical Support)	By the database server
ac_config.std	**$INFORMIXDIR/etc**, **%INFORMIXDIR%\etc**	Template for **archecker**-parameter values	By the database server
bar_act.log	**/tmp**, **%INFORMIXDIR%\etc**	ON–Bar activity log	By ON–Bar
bldutil.process_id	**/tmp**, **\tmp**	Error messages about the **sysutils** database appear in this file	By the database server
buildsmi.xxx	**/tmp**, **%INFORMIXDIR%\etc**	Error messages about SMI database	By the database server
concdr.sh	**$INFORMIXDIR /etc/conv**, **%INFORMIXDIR% \etc\conv**	Converts the **syscdr** database to Version 10.0 format	By the database server
.conf.dbservername		The **onsnmp** utility uses this file to obtain the database server configuration	By the database server
core	Directory from which the database server was invoked	Core dump	By the database server
Emergency boot files (For filenames, see page A-4.)	**$INFORMIXDIR/etc**, **%INFORMIXDIR%\etc**	Used in a cold restore	By ON–Bar
gcore (UNIX)	Specified by DUMPDIR configuration parameter	Assertion failure information	By the database server

Table A-1. List of Files That the Database Server Uses (continued)

Filename	Directory	Purpose	Created
illlsrra.xx	**$INFORMIXDIR/lib**, **%INFORMIXDIR%\lib**	Shared libraries for the database server and some utilities	By install procedure
.informix (UNIX)	User's home directory	Set personal environment variables	By the user
informix.rc (UNIX)	**$INFORMIXDIR/etc**	Set default environment variables for all users	By the database administrator
INFORMIXTMP	**/tmp**, **\tmp**	Temporary directory for internal files	By the database server
.inf.servicename	**/INFORMIXTMP,** *drive*:**\INFORMIXTMP**	Connection information	By the database server
.infos.dbservername	**$INFORMIXDIR/etc**, **%INFORMIXDIR%\etc**	Connection information	By the database server
.infxdirs	**/INFORMIXTMP,** *drive*:**\INFORMIXTMP**	Database server discovery file that **onsnmp** uses	By the database server
InstallServer.log (Windows)	**C:\temp**	Database server installation log	By the database server
ISM catalog	**$INFORMIXDIR/ism**, **%ISMDIR%**	Records saved backup objects and storage volumes that IBM Informix Storage Manager (ISM) uses	By ISM
ISM logs	**$INFORMIXDIR/ ism/logs**, **%ISMDIR%\logs**	Operator alert messages, backend status, additional ISM information	By ISM
ISMversion	**$INFORMIXDIR/ism**, **%ISMDIR%**	ISM version	During installation
JVM_*vpid*	Specified by JVPLOG configuration parameter	Messages that the Java virtual machine generates	By the Java virtual machine
JVPLOG	Specified by JVPLOG configuration parameter	Messages from the Java virtual processor	By the database server
.jvpprops	Specified by JVPPROFILE configuration parameter	Template for Java VP properties	During installation
Message log	Specified by MSGPATH configuration parameter	Error messages and status information	By the database server
The **ONCONFIG** file	**$INFORMIXDIR/etc**, **%INFORMIXDIR%\etc**	Configuration information	By the database administrator
onconfig	**$INFORMIXDIR/etc**, **%INFORMIXDIR%\etc**	Default ONCONFIG file (optional)	By the database server administrator
onconfig.std	**$INFORMIXDIR/etc**	Template for configuration-parameter values	During installation
oncfg_servername. servernum	**$INFORMIXDIR/etc**, **%INFORMIXDIR%\etc**	Configuration information for whole-system restores	By the database server
onsnmp.*servername*	**/tmp**, **\tmp**	Log file that the **onsnmp** subagent uses	By **onsnmp**
onsrvapd.log	**/tmp**, **\tmp**	Log file for the database server daemon **onsrvapd**	By **onsnmp**

Table A-1. List of Files That the Database Server Uses (continued)

Filename	Directory	Purpose	Created
revcdr.sh	**$INFORMIXDIR /etc/conv**, **%INFORMIXDIR% \etc\conv**	Reverts the **syscdr** database to an earlier format	By the database server
servicename.exp	**/INFORMIXTMP**, *drive*:**\INFORMIXTMP**	Connection information	By the database server
servicename.str	**/INFORMIXTMP**, *drive*:**\INFORMIXTMP**	Connection information	By the database server
shmem.xxx (UNIX)	Specified by DUMPDIR configuration parameter	Assertion-failure information	By the database server
sm_versions.std	**$INFORMIXDIR/etc**, **%INFORMIXDIR%\etc**	Identifies storage manager in use	During installation
snmpd.log	**/tmp**, **\tmp**	Log file for the SNMP master agent, **snmpdm**	By **onsnmp**
sqlhosts (UNIX)	**$INFORMIXDIR/etc**	Connection information; contained in the registry on Windows	During installation; modified by the database server administrator
VP**.servername.nnx**	**/INFORMIXTMP**, *drive*:**\INFORMIXTMP**	Connection information	By the database server
xbsa.messages	**$INFORMIXDIR /ism/applogs**, **%ISMDIR%\applogs**	XBSA library call information	By ISM

Descriptions of Files

This section provides short descriptions of the files listed in Table A-1.

af.xxx

The database server writes information about an assertion failure to the **af.xxx** file. The file is stored in the directory that the DUMPDIR configuration parameter specifies. For more information, see the information on monitoring for data inconsistency in your *IBM Informix Administrator's Guide*.

ac_msg.log

When you use **archecker** with ON–Bar to verify a backup, it writes brief status and error messages to the ON–Bar activity log and writes detailed status and error messages to the **archecker** message log (**ac_msg.log**). Technical Support uses the **archecker** message log to diagnose problems with backups and restores.

You specify the location of the **archecker** message log with the AC_MSGPATH configuration parameter. For more information, see the *IBM Informix Backup and Restore Guide*.

ac_config.std

The **ac_config.std** file contains the default **archecker** (archive checking) utility parameters. To use the template, copy it into another file, and modify the values. For a comprehensive list of the **archecker** parameters and how to use **archecker** with ON–Bar, see the *IBM Informix Backup and Restore Guide*.

bar_act.log

As ON–Bar backs up and restores data, it writes progress messages, warnings, and error messages to the ON–Bar activity log (**bar_act.log**). You specify the location of the ON–Bar activity log with the BAR_ACT_LOG configuration parameter. For more information, see the *IBM Informix Backup and Restore Guide*.

bldutil.process_id

If the database server cannot build the sysutils database, it creates the **bldutil.<process_id>** file which contains the error messages. The **process_id** value is the process ID of the **bldutil.sh** program. To access this output file, specify **${RESFILE}**.

buildsmi.xxx

If the database server cannot build the **sysmaster** database, it places a message in the message log that refers you to the **buildsmi.xxx** file. This file provides information about why the build failed. For information about the **sysmaster** database, refer to Chapter 2, "The sysmaster Database," on page 2-1.

concdr.sh

To convert the **syscdr** database from 7.31, 9.20, 9.21, 9.3 or 9.4 to 10.0 format, run the **concdr.sh** script on UNIX or the **concdr.bat** script on Windows. For details, see the *IBM Informix Migration Guide*.

.conf.dbservername

The **.conf.dbservername** file is created when you initialize the database server. The **onsnmp** utility queries this file to find out the configuration status of the database server. Do not delete this file.

The **.conf.dbservername** file contains information on shared memory and configuration that allows shared-memory clients to connect to the database server when they use utilities such as **onstat** or **onmode**.

core

The **core** file contains a core dump caused by an assertion failure. The database server writes this file to the directory from which the database server was invoked. For more information on monitoring for data inconsistency, see the chapter on consistency checking in the *IBM Informix Administrator's Guide*.

Emergency Boot Files for ON-Bar

The ON–Bar emergency boot files contain the information needed to perform a cold restore, and are updated after every backup. For details, see the *IBM Informix Backup and Restore Guide*.

The filename for the Dynamic Server emergency boot file is **ixbar_hostname.servernum**.

gcore.xxx (UNIX)

The database server writes information about an assertion failure to the **gcore.xxx** file. The file is stored in the directory specified by the DUMPDIR configuration parameter. For more information on monitoring for data inconsistency, see the chapter on consistency checking in the *IBM Informix Administrator's Guide*.

illlsrra.xx

The **illlsrra.xx** files are shared libraries that the database server and some database server utilities use. The shared libraries, if supported on your platform, are installed in **$INFORMIXDIR/lib** or **%INFORMIXDIR%\lib**.

The naming convention of the Informix shared library filename is as follows: i*lllsrra.xx*

lll	library class (for example, `asf` or `smd`)
s	library subclass (d=DSA; s=standard)
rr	major release number (for example, `07` or `08`)
a	library version ID (for example, `a` or `b`)
xx	shared-library filename extension (for example, `so`)

UNIX Only

Symbolic links to these files are automatically created in **/usr/lib** when the products are installed on your computer.

Important: The symbolic links to the shared libraries in **/usr/lib** are automatically created by the product installation procedures. However, if your **$INFORMIXDIR** is not installed using the standard installation method (for example, your **$INFORMIXDIR** is NFS-mounted from another computer), you or your system administrator might need to create manually the symbolic links of the shared libraries in **/usr/lib**.

End of UNIX Only

~/.informix

The **~/.informix** file is the *private-environment file*. Users can create this file and store it in their home directory. The *IBM Informix Guide to SQL: Reference* discusses the environment-configuration files.

informix.rc (UNIX)

The **/informix.rc** file is the *environment-configuration file*. You can use it to set environment variables for all users of IBM Informix products. The *IBM Informix Guide to SQL: Reference* discusses the environment-configuration files.

INFORMIXTMP

The **INFORMIXTMP** directory is an *internal database server directory*. During initialization, the database server creates this directory (if it does not exist yet) for storing internal files that must be local and relatively safe from deletion. The **onsnmp** utility uses the files in the **INFORMIXTMP** directory.

.inf.servicename

The database server creates the **.inf.servicename** file if any DBSERVERNAME or DBSERVERALIASES uses a shared-memory connection type. The database server removes the file when you take the database server offline. The name of this file is derived from the servicename field of the **sqlhosts** file or registry.

The database server keeps information about client/server connections in this file. You do not use the **.inf.servicename** file directly. You only need to recognize that it is a legitimate file when it appears in the **INFORMIXTMP** directory.

If this file is accidentally deleted, you must restart the database server.

.infos.dbservername

The database server creates the **.infos.dbservername** file when you initialize shared memory and removes the file when you take the database server offline. This file resides in **$INFORMIXDIR/etc** or **%INFORMIXDIR%\etc**. The name of this file is derived from the DBSERVERNAME parameter in the ONCONFIG configuration file.

The **.infos.dbservername** file contains information on shared memory and configuration that allows shared-memory clients to connect to the database server when they use utilities such as **onstat** or **onmode**. Do not delete this file.

.infxdirs

The database server maintains an **.infxdirs** file in the **INFORMIXTMP** directory. This file contains a line for every **INFORMIXDIR** from which a database server has been launched. If you remove the **.infxdirs** file, **onsnmp** cannot discover any database servers until the next time you restart the database server. Each time you restart the database server, it re-creates the **.infxdirs** file.

InstallServer.log (Windows)

The database server creates the **InstallServer.log** during installation.

ISM Catalog

ISM creates the ISM catalog during the **ism_startup** initialization. The ISM catalog records information about backup and restore save sets and about storage volumes that the storage manager uses. The ISM catalog records are stored in the **mm**, **index**, and **res** files in the **$INFORMIXDIR/ism** or **%ISMDIR%\ism** directory. For more information, see the *IBM Informix Storage Manager Administrator's Guide*.

ISM Logs

ISM creates several logs during ON–Bar backup and restore operations. The message window in the ISM Administrator GUI displays messages from these logs.

Log	Description
daemon.log	ISM backend status
messages	Operator alert messages
summary	Additional ISM information

For more information, see the *IBM Informix Storage Manager Administrator's Guide*.

ISMversion

The **ISMversion** file, which is installed with the database server, identifies the ISM version. Do not edit this file.

JVM_vpid

When the `0x10` bit is on for AFCRASH or the **AFDEBUG** environment variable is on, all the messages that the Java virtual machine generates are logged into the **JVM_vpid** file, where **vpid** is the process ID of the Java virtual processor. For more information, see *J/Foundation Developer's Guide*.

JVPLOG

When JVPDEBUG is set to 1, the database server writes tracing messages to the **JVPLOG** file. You can adjust the tracing level. On UNIX, you can have multiple **JVPLOG** files, one for each JVP virtual processor. On Windows, only one **JVPLOG** file exists. To obtain the JVP IDs, use the **onstat -g glo** command. For more information, see *J/Foundation Developer's Guide*.

.jvpprops

The **.jvpprops** file sets the Java virtual processor properties. Copy the **.jvpprops.template** to a new file named **.jvpprops**, and modify the values. For more information, see *J/Foundation Developer's Guide*.

Message Log

The database server writes status and error information to the message-log file. You specify the filename and location of the message log with the MSGPATH configuration parameter. For more information, refer to "MSGPATH" on page 1-53.

onconfig.std

The **onconfig.std** file serves as the template for creating the ONCONFIG configuration file. To use the template, copy it to another file and modify the values.

Important: Do not modify or delete **onconfig.std**. The database server uses values listed in this file when those values are missing from the ONCONFIG file.

For a comprehensive list of the ONCONFIG parameters, see Chapter 1, "Configuration Parameters," on page 1-1.

The ONCONFIG File

The *current configuration fil*e is the **%INFORMIXDIR%\etc\%ONCONFIG%** or **$INFORMIXDIR/etc/$ONCONFIG** file. The database server uses the ONCONFIG file during initialization.

If you start the database server with oninit and do not explicitly set the **ONCONFIG** environment variable, the database server looks for configuration values in the **onconfig.std** file. If no **onconfig.std** file exists, the database server returns the following error message:

```
WARNING: Cannot access configuration file $INFORMIXDIR/etc/$ONCONFIG.
```

For more information on the order of files where the database server looks for configuration values during initialization, refer the material on initializing the database server in the *IBM Informix Administrator's Guide*.

For more information on setting up your ONCONFIG file, refer to the materials on installing and configuring the database server in the *IBM Informix Administrator's Guide*.

onconfig

The **onconfig** file is an optional file that you create in the $**INFORMIXDIR/etc** or **%INFORMIXDIR%\etc** directory. The **onconfig** file is the default configuration file if the **ONCONFIG** environment variable is not set. For more information, refer to processing the configuration file in the *IBM Informix Administrator's Guide*.

To create the **onconfig** file, you can copy **onconfig.std** or one of your customized configuration files. For more information on setting up your **ONCONFIG** file, refer to installing and configuring the database server in the *IBM Informix Administrator's Guide*.

oncfg_servername.servernum

The database server creates the **oncfg_servername.servernum** file in the **$INFORMIXDIR/etc** or **%INFORMIXDIR%\etc** directory when you initialize disk space. The database server updates the file every time that you add or delete a dbspace, a logical-log file, or a chunk. The database server uses the **oncfg_servername.servernum** file when it salvages logical-log files during a whole-system restore. The database server derives the name of this file from the values of the DBSERVERNAME and SERVERNUM parameters in the ONCONFIG configuration file.

The database server uses the **oncfg_servername.servernum** files, so do not delete them. For more information, refer to creating the **oncfg_servername.servernum** file in the *IBM Informix Administrator's Guide* and the *IBM Informix Backup and Restore Guide*.

onsnmp.servername

The **onsnmp** subagent uses this log file. For more information, see the *IBM Informix SNMP Subagent Guide*.

This log file is called **onsnmp.servername** on Dynamic Server.

onsrvapd.log

The **onsrvapd** daemon uses this log file. For more information, see the *IBM Informix SNMP Subagent Guide*.

revcdr.sh

To revert the **syscdr** database from 10.0 to 9.4, 9.3, 7.31, 9.20, or 9.21 format, run the **revcdr.sh** script on UNIX or the **revcdr.bat** script on Windows. For details, see the *IBM Informix Migration Guide*.

shmem.xxx (UNIX)

The database server writes information about an assertion failure to the **shmem.xxx** file. The file is stored in the directory that the DUMPDIR configuration parameter specifies. For more information on monitoring for data inconsistency, see the chapter on consistency checking in the *IBM Informix Administrator's Guide*.

sm_versions.std

The **sm_versions.std** file is a template for the **sm_versions** file that you create. The **sm_versions** file contains a line identifying the current storage-manager version.

The storage manager uses the data in the **sm_versions** file (no **.std** suffix). To update the storage-manager version, edit the **sm_versions** file and then run the **ism_startup** command. For more information, see the *IBM Informix Backup and Restore Guide*.

snmpd.log

The SNMP master agent, **snmpdm** uses this log file. For more information, see the *IBM Informix SNMP Subagent Guide*.

sqlhosts

UNIX Only

The **sqlhosts** file is the *connectivity file* on UNIX platforms. It contains information that lets an IBM Informix client connect to an IBM Informix database server. For more information on the **sqlhosts** file, see client/server communications in the *IBM Informix Administrator's Guide*.

End of UNIX Only

Windows Only

On Windows, the connectivity information is in the **HKEY_LOCAL_MACHINE\SOFTWARE\INFORMIX\SQLHOSTS** key in the Windows registry.

End of Windows Only

VP.servername.nnx

The database server creates the VP**.servername.nnx** file, if needed, when you initialize shared memory. The name of this file comes from DBSERVERNAME or DBSERVERALIASES in the ONCONFIG file, the VP number (**nn**), and an internal identifier (**x**).

The database server keeps information about client/server connections in the VP**.servername.nnx** file. You do not use the file directly. You only need to recognize that it is a legitimate file.

If this file is accidentally deleted, you must restart the database server.

xbsa.messages

The **xbsa.messages** log contains XBSA library call information. ON–Bar and ISM use XBSA to communicate with each other. Technical Support would use the **xbsa.messages** log to diagnose problems with ON–Bar and ISM communications.

Appendix B. Trapping Errors

Occasionally, a series of events causes the database server to return unexpected error codes. If you do not have the appropriate diagnostic tools in place when these events occur, it might be difficult for you to determine the cause of these errors. This section discusses the following diagnostic tools:

- **onmode -I**
- tracepoints

Collecting Diagnostics using onmode -I

To help collect additional diagnostics, you can use **onmode -I** to instruct the database server to perform the diagnostics collection procedures that the *IBM Informix Administrator's Guide* describes. To use **onmode -I** when you encounter an error number, supply the *iserrno* and an optional session ID. The **-I** option is just one of many **onmode** options. For more information about **onmode**, see "In This Chapter" on page 10-1.

Syntax

```
                           ┌,──────┐
                           ▼       │
►►──onmode──-I──iserrno─────┬─────┬┴──────────────────────────────►◄
                           └─sid─┘
```

Element	Purpose	Key Considerations
-I ***iserrno***	Error number of the error for which you want to collect diagnostic information	None.
sid	Session ID of the session for which you want to collect diagnostic information	None.

Creating Tracepoints

Tracepoints are useful in debugging user-defined routines written in C. You can create a user-defined tracepoint to send special information about the current execution state of a user-defined routine.

Each tracepoint has the following parts:

- A *trace class* groups related tracepoints together so that they can be turned on or off at the same time.

  You can either use the built-in trace class called **_myErrors** or create your own. To create your own trace class, you insert rows into the **systraceclasses** system catalog table.
- A *trace message* is the text that the database server sends to the tracing-output file.

  You can store internationalized trace messages in the **systracemsgs** system catalog table.
- A *tracepoint threshold* determines when the tracepoint executes.

By default, the database server puts all trace messages in the trace-output file in the **tmp** directory with the following filename:

```
session_num.trc
```

For more information on tracing user-defined routines, see the *IBM Informix DataBlade API Programmer's Guide*.

Appendix C. Event Alarms

The database server provides a mechanism for automatically triggering administrative actions based on an event that occurs in the database server environment. This mechanism is the event-alarm feature. Events can be informative (for example, `Backup Complete`) or can indicate an error condition that requires your attention (for example, `Unable to Allocate Memory`).

Using ALARMPROGRAM to Capture Events

On UNIX, use the **alarmprogram.sh** and on Windows, use the **alarmprogram.bat** shell script, for handling event alarms and starting automatic log backups. For the setup instructions, see "ALARMPROGRAM" on page 1-12.

To automate logical-log backups only, two ready-made scripts are provided: **log_full.[sh|bat]** and **no_log.[sh|bat]**. Set ALARMPROGRAM to the full pathname of the script. For information, see "ALARMPROGRAM" on page 1-12.

Setting ALRM_ALL_EVENTS

You can set ALRM_ALL_EVENTS to specify whether ALARMPROGRAM runs for all events that are logged in the MSGPATH or only specified noteworthy events (events greater than severity 1).

Writing Your Own Alarm Script

Alternatively, you can write your own shell script, batch file, or binary program that contains the event-alarm parameters. When an event occurs, the database server invokes this executable file and passes it the event-alarm parameters (see Table C-1 on page C-2). For example, your script can use the **class_id** and **class_msg** parameters to take administrative action when a table failure occurs. Set ALARMPROGRAM to the full pathname of this executable file.

Customizing the ALARMPROGRAM scripts

Follow these steps to customize the **alarmprogram.[sh|bat]** script. You can use **alarmprogram.[sh|bat]** instead of **log_full.[sh|bat]** to automate log backups.

To customize the ALARMPROGRAM scripts:

1. Change the value of ADMINMAIL to the email address of the database server administrator.
2. Change the value of PAGERMAIL to the pager service email address.
3. Set the value of the parameter MAILUTILITY with **/usr/bin/mail** for UNIX and **$INFORMIXDIR/bin/ntmail.exe** for Windows.
4. To automatically back up logical logs as they fill, change BACKUP to `yes`.

   To stop automatic log backups, change BACKUP to any value other than `yes`.
5. In the ONCONFIG file, set ALARMPROGRAM to the full pathname of **alarmprogram.[sh|bat].**
6. Reboot the database server.

Alarms with a severity of 1 or 2 do not write any messages to the message log nor send email. Alarms with severity of 3 or greater send email to the database administrator. Alarms with severity of 4 and 5 also notify a pager via email.

Interpreting Error Messages

Some of the events that the database server reports to the message log cause it to invoke the alarm program. The class messages indicate the events that the database server reports.

The database server reports a nonzero exit code in the message log. In the alarm program, set the EXIT_STATUS variable to `0` for successful completion and to another number for a failure.

For example, if a thread attempts to acquire a lock, but the maximum number of locks that LOCKS specifies has already been reached, the database server writes the following message to the message log:

```
10:37:22  Checkpoint Completed:  duration was 0 seconds.
10:51:08  Lock table overflow - user id 30032, rstcb 10132264
10:51:10  Lock table overflow - user id 30032, rstcb 10132264
10:51:12  Checkpoint Completed:  duration was 1 seconds.
```

When the database server invokes **alarmprogram.[sh|bat]** or your alarm program, it generates a message that describes the severity and class of the event. If the severity is greater than 2, the message takes the following format:

```
Reasonably severe server event:
Severity: 3
Class ID: 21
Class msg: Database server resource overflow: 'Locks'.
Specific msg: Lock table overflow - user id 30032, rstcb 10132264
See Also:                          # optional message
The following message appears at the end of each e-mailed message:
This e-mail was generated by the server ALARMPROGRAM script on
 servername
 because something untoward just happened to eventname.
```

Event-Alarm Parameters

Table C-1 lists the event-alarm parameters.

Table C-1. Event-Alarm Parameters

Parameter	Meaning	Type
severity	Event severity (See Table C-2 for values.)	integer
class_id	Event class ID (See Table C-3 for values.)	integer
class_msg	Event class message (See Table C-3 for messages.)	string
specific_msg	Event specific messages	string
see_also	Event see-also file	string

Event Severity

The first parameter passed to the alarm program is the event-severity code. All events reported to the message log have one of the severity codes listed in Table C-2. Message-log events that have severity 1 do not cause the database server to invoke the alarm program unless the ALRM_ALL_EVENTS configuration parameter, supported by Dynamic Server, Version 10.0 or later, is enabled.

Table C-2. Event-Severity Codes

Severity	Description
1	Not noteworthy. The event (for example, date change in the message log) is not reported to the alarm program unless ALRM_ALL_EVENTS is enabled.
2	Information. No error has occurred, but some routine event completed successfully (for example, checkpoint or log backup completed).
3	Attention. This event does not compromise data or prevent the use of the system; however, it warrants attention (for example, one chunk of a mirrored pair goes down). Sends e-mail to the system administrator.
4	Emergency. Something unexpected occurred that might compromise data or access to data (assertion failure, or **oncheck** reports data corrupt). Take action immediately. Pages the system administrator.
5	Fatal. Something unexpected occurred and caused the database server to fail. Pages the system administrator.

Event Class ID

An event class ID is an integer that the database server substitutes as the second parameter in your alarm program. Each event class ID is associated with one of the events that causes the database server to run your alarm program.

Class Message

A class message is the text of the message that the database server substitutes for the third parameter of your alarm program when an event causes the database server to run your alarm program. The class messages are different for Dynamic Server and Extended Parallel Server.

Specific Messages

The database server substitutes additional information for the fourth parameter of your alarm program. In general, the text of this message is that of the message written to the message log for the event.

See Also Paths

For some events, the database server writes additional information to a file when the event occurs. The pathname in this context refers to the pathname of the file where the database server writes the additional information.

Event Alarms on Dynamic Server

Table C-3 on page C-3 shows the class IDs and class messages for alarms on Dynamic Server. The first column lists the class IDs that identify each alarm and the second column lists the class messages. For more information about setting the ALARMPROGRAM parameter, which controls alarms, see "ALARMPROGRAM" on page 1-12.

Table C-3. Event Alarms on Dynamic Server

Class ID	Class Message
1	Table failure: *'dbsname:"owner".tabname'*
2	Index failure: *'dbsname:"owner".tabname-idxname'*
3	Blob failure: *'dbsname:"owner".tabname'*
4	Chunk is offline, mirror is active: *chunk number*

Table C-3. Event Alarms on Dynamic Server (continued)

Class ID	Class Message
5	Dbspace is offline: *'dbspace name'*
6	Internal subsystem failure: *'message'*
7	Database server initialization failure
8	Physical restore failure
9	Physical recovery failure
10	Logical recovery failure
11	Cannot open chunk: *'pathname'*
12	Cannot open dbspace: *'dbspace name'*
13	Performance improvement possible
14	Database failure. *'database name'*
15	High-Availability Data-Replication failure
16	Backup completed: *'dbspace list'*
17	Backup aborted: *'dbspace list'*
18	Log backup completed: *log number*
19	Log backup aborted: *log number*
20	Logical logs are full—backup is needed
21	Database server resource overflow: *'resource name'*
22	Long transaction detected
23	Logical log *'number'* complete
24	Unable to allocate memory
25	Internal subsystem initialized: *'message'* (starts the optical subsystem)
26	Dynamically added log file logid
27	Log file required
28	No space for log file
N/A	Chunk (storage) failure
N/A	Data capacity
N/A	Logical log capacity
N/A	Maximum locks
N/A	Maximum capacity
N/A	Maximum sessions

Appendix D. Discontinued Configuration Parameters

This section lists the discontinued and obsolete configuration parameters for Dynamic Server.

Table D-1 summarizes the discontinued parameters. Although these parameters are still supported, it is recommended that you do not use them. Remove these parameters from the ONCONFIG file before using the VPCLASS parameter.

Table D-1. Discontinued Configuration Parameters

Configuration Parameter	Reference
AFF_NPROCS	page D-1
AFF_SPROC	page D-2
BUFFERS	page D-2
LRU_MAX_DIRTY	page D-4
LRU_MIN_DIRTY	page D-4
LRUS	page D-4
NOAGE	page D-5
NUMAIOVPS	page D-6
NUMCPUVPS	page D-6

Table D-2 summarizes the configuration parameters that are no longer supported.

Table D-2. Obsolete Configuration Parameters

Configuration Parameter	Reference
LBU_PRESERVE	page D-3
LOGSMAX	page D-3

AFF_NPROCS

onconfig.std *value* `0`

units Number of CPUs

range of values 0 through number of CPUs in the computer

takes effect When the database server shuts down and restarts

refer to The following material:

- Virtual-processor classes, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide*
- "AFF_SPROC" on page D-2
- "VPCLASS" on page 1-85

On multiprocessor computers that support *processor affinity*, AFF_NPROCS specifies the number of CPUs to which the database server can bind CPU virtual processors. Binding a CPU virtual processor to a CPU causes the virtual processor to run exclusively on that CPU. The database server assigns CPU virtual processors to CPUs in serial fashion, starting with the processor number that AFF_SPROC specifies.

If you specify more CPU virtual processors than there are processors, the database server starts over again at the beginning. For example, if you set AFF_NPROCS to 3 and AFF_SPROCS to 5, the database server assigns two CPU virtual processors to processor 5, two CPU virtual processors to processor 6, and one CPU virtual processor to processor 7.

Important: Use VPCLASS instead of AFF_NPROCS to specify the number of CPUs. You cannot use both AFF_NPROCS and VPCLASS *cpu* in the same ONCONFIG file.

AFF_SPROC

onconfig.std *value*	`0`
units	CPU number
range of values	0 through `(AFF_NPROCS - NUMCPUVPS + 1)`
takes effect	When the database server shuts down and restarts
refer to	The following material: • Virtual-processor classes, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide* • "AFF_NPROCS" on page D-1 • "VPCLASS" on page 1-85

On multiprocessor computers that support *processor affinity*, AFF_SPROC specifies the CPU, starting with 0, on which the database server starts binding CPU virtual processors to CPUs. The AFF_NPROCS parameter specifies the number of CPUs that the database server will use. The NUMCPUVPS parameter specifies the number of CPU virtual processors to be started, and the AFF_SPROC parameter specifies the CPU on which the first virtual processor is to start. For example, if you assign eight CPUs (AFF_NPROCS = `8`), and set NUMCPUVPS to `3` and AFF_SPROC to `5`, the database server binds CPU virtual processors to the fifth, sixth, and seventh CPUs.

Important: Use VPCLASS instead of AFF_SPROC to specify processor affinity. You cannot use both AFF_SPROC and VPCLASS *cpu* in the same ONCONFIG file.

BUFFERS

onconfig.std *value*	UNIX: `5000` Windows: `2000`
units	Number of buffers
range of values	For 32-bit platform on UNIX: with page size equal to 2048 bytes: 100 through 1,843,200 buffers `(1843200 = 1800 * 1024)`

	with page size equal to 4096 bytes: 100 through 921,600 buffers `(921,600 = ((1800 * 1024)/4096) * 2048 )` For 32-bit platform on Windows: 100 through 524,288 buffers (`524,288 = 512 * 1024`) For 64-bit platforms: 100 through 2^{31}-1 buffers (For the actual value for your 64-bit platform, see your machine notes. The maximum number of buffers on Solaris is 536,870,912.)
takes effect	When the database server is shut down and restarted
utilities	**onstat -b** or **-B** (See 14-8.)
refer to	The following material: • Shared-memory buffer pool in the shared-memory chapter of the *IBM Informix Administrator's Guide* • "RA_PAGES" on page 1-64 • "RA_THRESHOLD" on page 1-64 • Your *IBM Informix Performance Guide*

Note: Information that was specified with the BUFFERS configuration parameter prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter. For more information, see "BUFFERPOOL" on page 1-14.

BUFFERS specifies the maximum number of shared-memory buffers that the database server user threads have available for disk I/O on behalf of client applications. Therefore, the number of buffers that the database server requires depends on the applications. For example, if the database server accesses 15 percent of the application data 90 percent of the time, you need to allocate enough buffers to hold that 15 percent. Increasing the number of buffers can improve system performance.

In general, buffer space should range from 20 to 25 percent of physical memory. It is recommended that you calculate all other shared-memory parameters after you set buffer space (`BUFFERS` **system_page_size*) to 20 percent of physical memory.

LBU_PRESERVE

Dynamic Server no longer supports the LBU_PRESERVE parameter, which reserves the last logical log for ON–Archive use. ON–Archive, which has been discontinued, was the only utility that required free log space to back up a logical log.

LOGSMAX

Dynamic Server no longer supports the LOGSMAX parameter.

LOGSMAX specifies the maximum number of logical-log files for a database server instance. The database server requires at least three logical-log files for operation. The maximum number of logical logs is 32,767. The LOGSMAX value must be equal to or less than the highest log file number.

LRU_MAX_DIRTY

onconfig.std *value*	`60.00`
units	Percent
range of values	0 through 100 (fractional values are allowed)
takes effect	When the database server is shut down and restarted
refer to	The following topics in the shared-memory chapter of the *IBM Informix Dynamic Server Administrator's Guide* • LRU queues • Limiting the number of pages added to the MLRU queues

Note: Information that was specified with the LRU_MAX_DIRTY configuration parameter prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter. For more information, see "BUFFERPOOL" on page 1-14.

LRU_MAX_DIRTY specifies the percentage of modified pages in the LRU queues at which the queue is cleaned. If a parameter is specified out of the range of values, then the default of 60.00 percent is set.

LRU_MIN_DIRTY

onconfig.std *value*	`50.00`
units	Percent
range of values	0 through 100 (fractional values are allowed)
takes effect	When the database server is shut down and restarted
refer to	The following topics in the shared-memory chapter of the *IBM Informix Dynamic Server Administrator's Guide*: • LRU queues • When MLRU cleaning ends

Note: Information that was specified with the LRU_MIN_DIRTY configuration parameter prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter. For more information, see "BUFFERPOOL" on page 1-14.

LRU_MIN_DIRTY specifies the percentage of modified pages in the LRU queues at which page cleaning is no longer mandatory. Page cleaners might continue cleaning beyond this point under some circumstances. If a parameter is specified out of the range of values, then the default of 50.00 percent is set.

LRUS

onconfig.std *value*	`8`

if not present	If MULTIPROCESSOR is set: `MAX(4, num_cpu_vps)` If MULTIPROCESSOR is not set: `4`
units	Number of LRU queues
range of values	1 through 128
takes effect	When the database server is shut down and restarted
utilities	**onstat -R** (See page 14-90.)
refer to	The following material: • LRU queues, in the shared-memory chapter of the *IBM Informix Dynamic Server Administrator's Guide* • Chapter on configuration effects on memory, in your *IBM Informix Dynamic Server Performance Guide*

Note: Information that was specified with the LRUS configuration parameter prior to Version 10.0 is now specified using the BUFFERPOOL configuration parameter. For more information, see "BUFFERPOOL" on page 1-14.

LRUS specifies the number of LRU (least-recently-used) queues in the shared-memory buffer pool. You can tune the value of LRUS, in combination with the LRU_MIN_DIRTY and LRU_MAX_DIRTY parameters, to control how frequently the shared-memory buffers are flushed to disk.

Setting LRUS too high might result in excessive page-cleaner activity.

NOAGE

onconfig.std *value*	`0`
range of values	`0` = Use priority aging. `1` = Disable priority aging.
takes effect	When the database server shuts down and restarts
refer to	The following material: • Preventing priority aging, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide* • "VPCLASS" on page 1-85

Some operating systems lower the priority of processes as the processes run over a long period of time. NOAGE, when set to `1`, disables *priority aging* of CPU virtual processors by the operating system. When NOAGE is set to the default of `0`, the operating system might lower the priority of CPU virtual processors, as well as other processes, as they accumulate processing time. If your operating system supports priority aging, it is recommended that you set NOAGE to 1.

Important: It is recommended that you specify priority aging with the VPCLASS parameter instead of the NOAGE parameter. You cannot use both NOAGE and VPCLASS *cpu* in the same ONCONFIG file.

NUMAIOVPS

onconfig.std ***value***	None
if not present	(2 * *number_of_chunks*) or 6, whichever is greater; *number_of_chunks* is the number of chunks that you have allocated.
units	Number of AIO VPs
range of values	Integer greater than or equal to 1
takes effect	When the database server shuts down and restarts
utilities	**onmode -p** in "Add or Remove Virtual Processors" on page 10-12
refer to	The following material: • Asynchronous I/O, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide* • "VPCLASS" on page 1-85

NUMAIOVPS specifies the number of virtual processors of the AIO class to run. Unless kernel asynchronous I/O is implemented, the AIO virtual processors perform all the database server disk I/O, other than I/O to the log files.

Important: It is recommended that you specify the number of AIO VPs with VPCLASS *aio* instead of NUMAIOVPS. You cannot use both NUMAIOVPS and VPCLASS *aio* in the same ONCONFIG file.

UNIX Only

If your platform has kernel-asynchronous I/O (KAIO) turned on, the database server uses AIO virtual processors to perform I/O only to cooked chunks. The database server uses KAIO to perform all I/O to raw disk space and to the physical and logical logs. For details, see the machine notes.

End of UNIX Only

NUMCPUVPS

onconfig.std ***value***	1
units	Number of CPU VPs
range of values	1 through the number of CPUs
takes effect	When the database server shuts down and restarts
utilities	**onmode -p** in "Add or Remove Virtual Processors" on page 10-12
refer to	The following material: • CPU virtual processors, in the chapter on virtual processors and threads in the *IBM Informix Administrator's Guide* • "VPCLASS" on page 1-85

NUMCPUVPS specifies the number of virtual processors of the CPU class to run. CPU virtual processors run all threads that start as the result of a connection by a client application, as well as internal threads. In general, allocate only one CPU virtual processor on a single-processor computer or node. On a multiprocessor computer or node, do not allocate more CPU virtual processors than there are CPUs.

Important: It is recommended that you specify the number of CPU virtual processors with VPCLASS *cpu* instead of NUMCPUVPS. You cannot use both NUMCPUVPS and VPCLASS *cpu* in the same ONCONFIG file.

On UNIX, use the **onmode -p -1 CPU** command to decrease the number of CPU VPs. On Windows, you can add a CPU VP, but you cannot subtract it.

Appendix E. Error Messages

This chapter lists nonnumbered messages that are printed in the database server message log and provides corrective actions.

For information on numbered messages and the unnumbered ON–Bar messages, search for the message text in the error messages file, which is located in the subdirectory for your locale under the **$INFORMIXDIR/msg** directory. You can also search *IBM Informix Error Messages* in English at the IBM Informix Online Documentation site at http://www.ibm.com/software/data/informix/pubs/library/.

Some of the messages included below might require you to contact Technical Support staff. Such messages are rarely, if ever, seen at customer locations.

For information on what the message log is, see installing and configuring the database server in the *IBM Informix Administrator's Guide*. For information on specifying the path to the message file, see "MSGPATH" on page 1-53.

How the Messages Are Ordered in This Chapter

Database server message-log messages are arranged in this chapter in alphabetical order, sorted with the following additional rules:

- The time stamp that precedes each message is ignored.
- Letter case is ignored in alphabetization.
- Spaces are ignored.
- Quotation marks are ignored.
- Leading ellipses are ignored.
- The word *the* is ignored if it is the first word in the message.
- Messages that begin with numbers or punctuation symbols appear toward the end of the list in a special section labeled "Messages: Symbols" on page E-20.
- Certain related messages are grouped together, as follows:
  - "Conversion/Reversion Messages" on page E-21
  - "Conversion and Reversion Messages for Enterprise Replication" on page E-26
  - "Dynamic Log Messages" on page E-27
  - "Sbspace Metadata Messages" on page E-28
  - "Truncate Table Messages" on page E-29

A cause and suggested corrective action for a message or group of messages follow the message text.

How to View These Messages

Use one of the following methods to view these messages:

- Online message log

  To see the messages displayed as they occur, use the **tail -f** *online.log* command.
- **onstat -m** command

  For more information, see "onstat -m" on page 14-84.

- IBM Informix Server Administrator (ISA)

  For more information, see the ISA online help.

To see the error number associated with these unnumbered messages, view the **logmessage** table in the **sysmaster** database:

```
SELECT * FROM logmessage;
```

Message Categories

Four general categories of unnumbered messages exist, although some messages fall into more than one category:

- Routine information
- Assertion-failed messages
- Administrative action needed
- Fatal error detected

Technical Support uses the assertion-failed messages to assist in troubleshooting and diagnostics. The information that they report often falls into the category of *unexpected events* that might or might not develop into problems caught by other error codes. Moreover, the messages are terse and often extremely technical. They might report on one or two isolated statistics without providing an overall picture of what is happening. This information can suggest to technical support possible research paths.

Messages: A-B

Aborting Long Transaction: *tx 0xn.*

Cause: The transaction spans the log space specified by transaction high-watermark (LTXHWM), and the offending long transaction is rolling back.

Action: No additional action is needed. The address of the transaction structure in shared memory is displayed as a hexadecimal value.

Affinitied VP *mm* **to phys proc** *nn***.**

Cause: The database server successfully bound a CPU virtual processor to a physical processor.

Action: None required.

Affinity not enabled for this server.

Cause: You tried to bind your CPU virtual processors to physical processors, but the database server that you are running does not support process affinity.

Action: Set AFF_NPROCS to `0`, or remove the affinity setting from VPCLASS.

Assert Failed: Error from SBSpace cleanup thread.

Cause: The sbspace cleanup thread encountered an error while cleaning up stray smart large objects.

Action: See the action suggested in the message log file.

Most of the time, running **onspaces -cl sbspacename** on the failed sbspace succeeds in cleaning up any stray smart large objects. If you encounter an unrecoverable error, contact Technical Support.

Assert Failed: Short description of what failed
Who: Description of user/session/thread running at the time
Result: State of the affected database server entity
Action: What action the database administrator should take
See Also: DUMPDIR/af.uniqid containing more diagnostics.

Cause: This message indicates an internal error.

Action: The **af.uniqid** file in the directory specified by the ONCONFIG parameter DUMPDIR contains a copy of the assertion-failure message that was sent to the message log, as well as the contents of the current, relevant structures and/or data buffers. The information included in this message is intended for Technical Support.

Begin re-creating indexes deferred during recovery.

Cause: During recovery, indexes to be created are deferred until after recovery completes. This message indicates that the database server deferred re-creating indexes and that it is now creating the indexes. During the time that the database server re-creates the indexes, it locks the affected tables with a shared lock.

Action: None required.

Building 'sysmaster' database requires ~*mm* pages of logical log. Currently there are *nn* pages available. Prepare to back up your logs soon.

Cause: You do not currently have the approximate amount of free log space necessary to complete a build of the sysmaster database.

Action: Back up your logs.

Building 'sysmaster' database...

Cause: The database server is building the **sysmaster** database.

Action: None required.

Messages: C

Cannot Allocate Physical-log File, *mm* wanted, *nn* available.

Cause: The database server attempted to initialize shared memory with a physical-log size that exceeds the amount of contiguous space available in the dbspace (specified as PHYSDBS in ONCONFIG). Both quantities of space, wanted and available, are expressed as kilobytes.

Action: You must either reduce the size of the physical log (specified as PHYSFILE in ONCONFIG) or change the location of the physical log to a dbspace that contains adequate contiguous space to accommodate the physical log.

Cannot alter a table which has associated violations table.

Cause: The user tried to add, drop, or modify a column in a table that has a violations table associated with it.

Action: Do not change the columns in the user table.

Cannot change to mode.

Cause: Some error during fast or full recovery has prevented the system from changing to online or quiescent mode.

Action: See previous messages in the log file for information.

Cannot Commit Partially Complete Transactions.

Cause: Transactions that drop tables or indexes do not perform the drop until a COMMIT statement is processed (with a few exceptions). In these cases, a *beginning commit* log record is written, followed by the usual commit log record. If the database server fails in between the two, the fast recovery process attempts to complete the commit the next time that you initialize the database server.

If this completion of the commit fails, the database server generates the preceding message.

Action: To determine if you need to take action, examine the logical log as described in Chapter 4, "Interpreting Logical-Log Records," on page 4-1.

Cannot create a user-defined VP class with 'SINGLE_CPU_VP' non-zero.

Cause: SINGLE_CPU_VP is set to nonzero, and **onmode** was used to create a user-defined VP class.

Action: If user-defined VP classes are necessary, stop the database server, change SINGLE_CPU_VP to zero, and restart the database server.

Cannot create violations/diagnostics table.

Cause: The user issued a START VIOLATIONS TABLE statement for a target table. The database server cannot create the violations table for this target table. Any of the following situations might be the reason for this failure:

- The target table already has a violations table.
- You specified an invalid name for the violations table in the START VIOLATIONS TABLE statement. For example, if you omit the USING clause from the statement and if the number of characters in the target table plus four characters is longer than the maximum identifier length, the generated names of the violations table exceed the maximum identifier length.
- You specified a name for the violations table in the START VIOLATIONS TABLE statement that match the names of existing tables in the database.
- The target table contains columns with the names **informix_tupleid**, **informix_optype**, or **informix_recowner**. Because these column names duplicate the **informix_tupleid**, **informix_optype**, or **informix_recowner** columns in the violations table, the database server cannot create the violations table.
- The target table is a temporary table.
- The target table is serving as a violations table for some other table.
- The target table is a system catalog table.

Action: To resolve this error, perform one of the following actions:

- If the violations table name was invalid, specify a unique name for the violations table in the USING clause of the START VIOLATIONS TABLE statement.
- If the target table contains columns with the names **informix_tupleid**, **informix_optype**, or **informix_recowner**, rename them to something else.
- Choose a permanent target table that is not a system catalog table or a violations table for some other table.

Cannot insert from the violations table to the target table.

Cause: The user has issued a statement that attempts to insert rows from the violations table into the target table. For example, the user enters the following invalid statement:

```
INSERT INTO mytable SELECT * FROM mytable_vio;
```

Also, if the target table has filtering-mode constraints, you receive this error. Extended Parallel Server does not support filtering-mode constraints.

Action: To recover from this error, perform the following actions:

- Do not use filtering constraints.
- Stop the violations table.
- Insert rows from the violations table into a temporary table, and then insert rows from the temporary table into the target table.

Cannot modify/drop a violations/diagnostics table.

Cause: The user has tried to alter or drop a table that is serving as a violations table for another table.

Action: Do not alter or drop the violations table.

Cannot Open Dbspace ***nnn*****.**

Cause: The database server is unable to access the specified dbspace. This message indicates a problem opening the tblspace or corruption in the initial chunk of the dbspace.

Action: Verify that the device or devices that make up the chunks of this dbspace are functioning properly and that you assigned them the correct operating-system permissions (rw-rw----). You might be required to perform a data restore.

Cannot Open Logical Log.

Cause: The database server is unable to access the logical-log files. Because the database server cannot operate without access to the logical log, you must resolve this problem.

Action: Verify that the chunk device where the logical-log files reside is functioning and has the correct operating-system permissions (rw-rw----).

Cannot Open Mirror Chunk *pathname***, errorno =** *nn***.**

Cause: The database server cannot open the mirrored chunk of a mirrored pair. The chunk *pathname* and the operating-system error are returned.

Action: For more information about corrective actions, see your operating-system documentation.

Cannot Open Primary Chunk *pathname***, errorno =** *nnn***.**

Cause: The primary chunk of a mirrored pair cannot be opened. The chunk *pathname* and the operating-system error are returned.

Action: For more information about corrective actions, see your operating-system documentation.

Cannot Open Primary Chunk *chunkname.*

Cause: The *initial* chunk of the dbspace cannot be opened.

Action: Verify that the chunk device is running properly and has the correct operating-system permissions (rw-rw----).

Cannot open sysams in database *name***, iserrno** *number***.**

Cause: An error occurred when the database server opened the **sysams** system table.

Action: Note the error *number* and contact Technical Support.

Cannot open sysdistrib in database *name***, iserrno** *number***.**

Cause: An error occurred when the database server accessed the **sysdistrib** system table.

Action: Note the error *number* and contact Technical Support.

Cannot open *system_table* **in database** *name***, iserrno** *number***.**

Cause: An error occurred when the database server opened the specified system table.

Action: Note the error *number* and contact Technical Support.

Cannot open systrigbody in database *name***, iserrno** *number***.**

Cause: An error occurred when the database server accessed the **systrigbody** system table.

Action: Note the error *number* and contact Technical Support.

Cannot open systriggers in database *name*, iserrno *number*.

Cause: An error occurred when the database server accessed the **systriggers** system table.

Action: Note the error *number* and contact Technical Support.

Cannot open sysxtdtypes in database *name*, iserrno *number*.

Cause: An error occurred while accessing the **sysxtdtypes** system table.

Action: Note the error *number* and contact Technical Support.

Cannot Perform Checkpoint, shut system down.

Cause: A thread that is attempting to restore a mirrored chunk has requested a checkpoint, but the checkpoint cannot be performed.

Action: Shut down the database server.

Cannot Restore to Checkpoint.

Cause: The database server is unable to recover the physical log and thus unable to perform fast recovery.

Action: If the database server does not come online, perform a data restore from dbspace backup.

Cannot Rollback Incomplete Transactions.

Cause: Within the fast-recovery or data-restore procedure, the logical-log records are first rolled forward. Then, open transactions that have not committed are rolled back. An open transaction could fail during the rollback, leaving some of the modifications from the open transaction in place. This error does not prevent the database server from moving to quiescent or online mode, but it might indicate an inconsistent database.

Action: To determine if any action is needed, use the onlog utility to examine the logical log.

Cannot update pagezero.

Cause: A failure occurred while the database server was trying to rewrite a reserved page during the reversion process.

Action: See previous messages in the log file for information, or contact Technical Support.

Cannot update syscasts in database *name*. Iserrno *number*.

Cause: An internal error occurred while inserting data into the **syscasts** system table.

Action: Contact Technical Support..

Can't affinity VP *mm* to phys proc *nn*.

Cause: The database server supports process affinity, but the system call to bind the virtual processor to a physical processor failed.

Action: See your operating-system documentation.

Changing the sbspace minimum extent value: old value *value1*, new value *value2*.

Cause: This informational message occurs when you issue the following command:

```
onspaces -ch sbspace -Df "MIN_EXT_SIZE=value1" -y
```

Action: None. For more information, see "Change Sbspace Default Specifications" on page 13-14.

Checkpoint blocked by down space, waiting for override or shutdown.

Cause: A dbspace has gone down during a checkpoint interval. The database server is configured to wait for an override when this situation occurs.

Action: Either shut down the database server or issue an **onmode -O** command to override the down dbspace. For more information on the **onmode** utility, see "In This Chapter" on page 10-1.

Checkpoint Completed: duration was *n* seconds.

Cause: A checkpoint completed successfully.

Action: None required.

Checkpoint Page Write Error.

Cause: The database server detected an error in an attempt to write checkpoint information to disk.

Action: For additional assistance in resolving this situation, contact Technical Support.

Checkpoint Record Not Found in Logical Log.

Cause: The logical log or the chunk that contains the logical log is corrupted. The database server cannot initialize.

Action: Perform a data restore from dbspace backup.

Chunk *chunkname* **added to space** *spacename.*

Cause: The variables in this message have the following values:

chunkname is the name of the chunk that the database server administrator is adding.

spacename is the name of the storage space to which the database server administrator is adding the chunk.

Action: None required.

Chunk *chunkname* **dropped from space** *spacename.*

Cause: The database server administrator dropped chunk *chunkname* from space *spacename*.

Action: None required.

Chunk *number nn pathname* **-- Offline.**

Cause: The indicated chunk in a mirrored pair has been marked with status `D` and taken offline. The other chunk in the mirrored pair is operating successfully.

Action: Take steps now to repair the chunk device and restore the chunk. The chunk *number* and chunk device *pathname* are displayed.

Chunk *number nn pathname* **-- Online.**

Cause: The indicated chunk in a mirrored pair has been recovered and is online (marked with status `O`). The chunk *number* and chunk device *pathname* are displayed.

Action: None required.

The chunk *pathname* **must have READ/WRITE permissions for owner and group.**

Cause: The chunk *pathname*does not have the correct owner and group permissions.

Action: Make sure that you assigned the correct permissions
(-rw-rw---) to the device on which the chunk is located.

The chunk *pathname* **must have** *owner-ID* **and** *group-ID* **set to informix.**

Cause: The chunk *chunkname* does not have the correct owner and group ID.

Action: Make sure the device on which the chunk is located has the ownership. On UNIX, both owner and group should be **informix**. On Windows, the owner must be a member of the **Informix-Admin** group.

The chunk *pathname* **will not fit in the space specified.**

Cause: The chunk *pathname* does not fit in the space that you specified.

Action: Choose a smaller size for the chunk, or free space where the chunk is to be created.

Cleaning stray LOs in sbspace *sbspacename***.**

Cause: The database server administrator is running **onspaces -cl sbspacename**.

Action: None required.

Completed re-creating indexes.

Cause: The database server finished re-creating the deferred indexes.

Action: None required.

Configuration has been grown to handle up to *integer* **chunks.**

Cause: The database server administrator increased the number of chunks to the specified value by changing CONFIGSIZE or setting MAX_CHUNKS to a higher value.

Action: None required. The change was successful.

Configuration has been grown to handle up to *integer* **dbslices.**

Cause: The database server administrator increased the number of dbslices to the specified value by changing CONFIGSIZE or setting MAX_DBSLICES to a higher value.

Action: None required. The change was successful.

Configuration has been grown to handle up to *integer* **dbspaces.**

Cause: The database server administrator increased the number of dbspaces to the specified value by changing CONFIGSIZE or setting MAX_DBSPACES to a higher value.

Action: None required. The change was successful.

Continuing Long Transaction (for COMMIT): *tx 0xn***.**

Cause: The logical log has filled beyond the long-transaction high-watermark (LTXHWM), but the offending long transaction is in the process of committing. In this case, the transaction is permitted to continue writing to the logical log and is not rolled back. The address of the transaction structure in shared memory is displayed as hexadecimal value *tx 0xn*.

Action: None required.

Could not disable priority aging: errno = *number***.**

Cause: An operating-system call failed while it was trying to disable priority aging for the CPU virtual processor. The system error *number* associated with the failure is returned.

Action: See your operating-system documentation.

Could not fork a virtual processor: errno = *number***.**

Cause: The fork of a virtual processor failed. The database server returns the operating-system error *number* associated with the failure.

Action: For information on determining the maximum number of processes available per user and for the system as a whole, refer to your operating-system documentation.

Create_vp: cannot allocate memory.

Cause: The database server cannot allocate new shared memory.

Action: The database server administrator must make more shared memory available. This situation might require increasing SHMTOTAL or reconfiguring the operating system. This message is usually accompanied by other messages that give additional information.

Messages: D-E-F

Dataskip is OFF for all dbspaces.

Cause: Informational.

Action: None required.

Dataskip is ON for all dbspaces.

Cause: Informational.

Action: None required.

Dataskip is ON for dbspaces: *dbspacelist***.**

Cause: Informational; DATASKIP is ON for the specified dbspaces.

Action: None required.

Dataskip will be turned {ON|OFF} for *dbspacename***.**

Cause: Informational; DATASKIP is ON or OFF for the specified dbspace.

Action: None required.

DBSERVERALIASES exceeded the maximum limit of 32

Cause: The limit of 32 aliases was reached.

Action: Nothing. Only the first 32 will be used.

DBSPACETEMP internal list not initialized, using default.

Cause: An error occurred while initializing a user-specified DBSPACETEMP list. Typically this condition is due to a memory-allocation failure.

Action: Check for accompanying error messages.

The DBspace/BLOBspace *spacename* **is now mirrored.**

Cause: You successfully added mirroring to the indicated storage space.

Action: None required.

The DBspace/BLOBspace *spacename* **is no longer mirrored.**

Cause: You have ended mirroring for the indicated storage space.

Action: None required.

Dbspace *dbspacename* **for Physical-log File not found.**

Cause: The dbspace *dbspacename* specified by the PHYSDBS configuration parameter does not exist. As a consequence, the database server cannot complete initialization.

Action: Use a dbspace known to exist.

*devname***: write failed, file system is full.**

Cause: Because the file system *devname* is full, the write failed.

Action: Free some space in *devname*.

Dropping temporary tblspace *0xn***, recovering** *nn* **pages.**

Cause: During shared-memory initialization, the database server routinely searches for temporary tables that are left without proper cleanup. If the database server finds a temporary table, it drops the table and recovers the space. The database server located the specified temporary tblspace and dropped it. The value *0xn* is the hexadecimal representation of the tblspace number.

Action: None required.

Dynamically allocated new shared memory segment (size *nnnn*).

Cause: This status message informs you that the database server successfully allocated a new shared-memory segment of size *nnnn*.

Action: None required.

ERROR: NO "wait for" locks in Critical Section.

Cause: The database server does not permit a thread to own locks that might have to wait while that thread is within a critical section. Any such lock request is denied, and an ISAM error message is returned to the user.

Action: The error reported is an internal error. Contact IBM Informix Technical Support.

Error building sysmaster database. See *outfile*.

Cause: Errors were encountered in building the sysmaster database. The file *outfile* contains the result of running the script buildsmi.

Action: See the file *outfile*.

Error in dropping system defined type.

Cause: An internal error occurred while updating either the **sysxtdtypes**, **sysctddesc**, or **sysxtdtypeauth** system table.

Action: Contact Technical Support.

Error in renaming systdist.

Cause: An internal error occurred while trying to find and rename the **Informix.systdist** SPL routine.

Action: Contact Technical Support.

Error removing sysdistrib row for tabid = *tabid*, colid = *colid* in database *name*. iserrno = *number*

Cause: An error occurred while updating the **sysdistrib** system table.

Action: Note the error *number* and contact Technical Support.

Error writing *pathname* errno = *number*.

Cause: The operating system cannot write to *pathname. Number* is the number of the operating-system error that was returned.

Action: Investigate the cause of the operating-system error. Usually it means that no space is available for the file. It might also mean that the directory does not exist or that no write permissions exist.

Error writing shmem to file *filename* (*error*).
Unable to create output file *filename* errno=*mm.*
Error writing *filename* errno=*nn*.

Cause: The database server detected an error in an attempt to write shared memory to *filename*. The first message is followed by one of the next two. Either the attempt failed because the output file could not be created or because the contents of shared memory could not be written. The error refers to the operating-system error that prompted the attempted write of shared memory to a file. The value of *nn* is the operating-system error.

Action: See your operating-system documentation.

Fail to extend physical log space.

Cause: The attempt to extend the physical log space failed. Either the path does not exist or the permissions are incorrect.

Action: Use a path that exists. Check permissions on the current working directory. You or the system administrator must give your group execute permission on the current working directory. After your group has been given permission, retry the operation that generated this message.

Fatal error initializing CWD string.
Check permissions on current working directory.
Group *groupname* must have at least execute permission on '.'.

Cause: Group *groupname* does not have execute permission for the current working directory.

Action: Check permissions on the current working directory. You or the system administrator must give your group execute permission on the current working directory. After your group has been given permission, retry the operation that generated this message.

The following tables have outstanding old version data
pages due to an In-Place Alter Table. Perform UPDATE *tablename* SET column = *column* WHERE 1=1; to clear these pages from the following tables.

Cause: Reversion to a previous version of the database server has been attempted while an in-place ALTER TABLE is in progress. The previous versions of the database server cannot handle tables that have multiple schemas of rows in them.

Action: Force any in-place alters to complete by updating the rows in the affected tables before you attempt to revert to a previous version of the database server. To do this, create a dummy update in which a column in the table is set to its own value, forcing the row to be updated to the latest schema in the process without actually changing column values. Rows are always altered to the latest schema, so a single pass through the table that updates all rows completes all outstanding in-place alters.

Fragments *dbspacename1 dbspacename2* **of table** *tablename* **set to non-resident.**

Cause: The specified fragments of *tablename* either have been set to nonresident by the SET TABLE statement.

Action: None required.

Forced-resident shared memory not available.

Cause: The database server port for your computer does not support forced-resident shared memory.

Action: None required.

Freed *mm* **shared-memory segment(s)** *number* **bytes.**

Cause: The database server sends this message to the message log after you run the **-F** option of the **onmode** utility to free unused memory. The message informs you of the number of segments and bytes that the database server successfully freed.

Action: None required.

Messages: G-H-I

gcore *pid*; **mv core.***pid* **dir/core.***pid***.ABORT.**

Cause: This status message during a database server failure provides the name and place of each core file associated with the virtual processors.

Action: None required.

I/O *function* **chunk** *mm*, **pagenum** *nn*, **pagecnt** *aa* **--> errno =** *bb*.

Cause: An operating-system error occurred during an attempt to access data from disk space. The operating-system function that failed is defined by *function*. The chunk number and physical address of the page where the error occurred are displayed as integers. The *pagecnt* value refers to the number of pages that the thread was attempting to read or write. If an *errno* value is displayed, it is the number of the operating-system error and might explain the failure. If *function* is specified as *bad request*, some unexpected event caused the I/O attempt on an invalid chunk or page.

Action: If the chunk status changes to `D`, or down, restore the chunk from its mirror or repair the chunk. Otherwise, perform a data restore.

I/O error, *primary/mirror* **Chunk** *pathname* **-- Offline (***sanity***).**

Cause: The database server detected an I/O error on a primary or mirror chunk with *pathname*. The chunk was taken offline.

Action: Check that the device on which the chunk was stored is functioning as intended.

Deleted Indexes idx1 and idx 2 error message

Informix *database_server* **Initialized - Complete Disk Initialized.**

Cause: Disk space and shared memory have been initialized. Any databases that existed on the disk before the initialization are now inaccessible.

Action: None required.

Informix *database_server* **Initialized - Shared Memory Initialized.**

Cause: Shared memory has been initialized.

Action: None required.

Informix *database_server* **Stopped.**

Cause: The database server has moved from quiescent mode to offline mode. The database server is offline.

Action: None required.

ERROR: Insufficient available disk in the root dbspace to increase the entire Configuration save area.

Cause: The user attempted to increase the number of storage objects to a specific value by changing CONFIGSIZE or setting MAX_DBSPACES, MAX_DBSLICES, or MAX_CHUNKS to a higher value, but the database server did not have enough rootspace for the increased number of storage objects. A storage object might be a dbspace, dbslice, or chunk.

Action: Increase the size of the root dbspace or reset CONFIGSIZE, MAX_DBSPACES, MAX_DBSLICES, or MAX_DBSLICES to a lower value and restart the database server. For example, if you set MAX_CHUNKS to `32,768`, but the root dbspace did not have enough space, set MAX_CHUNKS to a lower value.

Insufficient available disk in the root dbspace for the CM save area. Increase the size of the root dbspace in the ONCONFIG file and reinitialize the server.

Cause: The cause might be one of the following:

- The user attempted to increase the number of storage objects to a specific value by changing CONFIGSIZE or setting MAX_DBSPACES, MAX_DBSLICES, or MAX_CHUNKS to a higher value, but the database server did not have enough rootspace for the

increased number of storage objects. A storage object might be a dbspace, dbslice, or chunk.
- The user converted to a database server version that requires slightly more rootspace, but it is not available (this case is unlikely).

Action: Take one of the following actions:
- Increase the size of the root dbspace or reset CONFIGSIZE, MAX_DBSPACES, MAX_DBSLICES, or MAX_DBSLICES to a lower value and restart the database server. For example, if you set MAX_DBSPACES to `32,768` but the root dbspace did not have enough space, set MAX_DBSPACES to a lower value.
- Increase the size of the root dbspace and reinitialize the database server.

Internal overflow of shmid's, increase system max shared memory segment size.

Cause: The database server was initializing shared memory when it ran out of internal storage for the shared-memory IDs associated with this segment.

Action: Increase the value of your maximum kernel shared-memory segment size, usually SHMMAX. For more information, see your operating-system documentation.

Messages: J-K-L-M

Listener-thread err = *error_number***:** *error_message.*

Cause: A listener thread has encountered an error. This message displays the error number and message text.

Action: For the cause and corrective action, see the IBM Informix Online Documentation site at http://www.ibm.com/software/data/informix/pubs/library/.

Lock table overflow - user id *mm* **session id** *nn.*

Cause: A thread attempted to acquire a lock when no locks were available. The user ID and session ID are displayed.

Action: Increase the LOCKS configuration parameter, and initialize shared memory.

Logical-log File not found.

Cause: The checkpoint record in the root dbspace reserved page is corrupted.

Action: Perform a data restore from dbspace backup.

Logical Log *nn* **Complete.**

Cause: The logical-log file identified by log-ID number *nn* is full. The database server automatically switches to the next logical-log file in the sequence.

Action: None required.

Logical logging *vberror* **for** *type:subtype* **in (***failed_system***).**

Cause: Logging failed. The log record that caused the error is identified as follows:

type	Is the logical-log record type.
subtype	Is the logging subsystem.
failed_system	Is the name of an internal function that indicates what system failed to log.

Action: Contact Technical Support.

Log Record: log = *ll***, pos =** *0xn***, type =** *type:subtype(snum),* **trans =** *xx*

Cause: The database server detected an error during the rollforward portion of fast recovery or logical-log restore.

The log record that caused the error is identified as follows:

ll	Is the logical-log ID where the record is stored.
0xn	Is the hexadecimal address position within the log.
type	Is the logical-log record type.
subtype	Is the logging subsystem.
snum	Is the subsystem number.
xx	Is the transaction number that appears in the logical log.

Action: Contact Technical Support.

Log record (*type:subtype***) at log** *nn***,** *0xn* **was not undone.**

Cause: A log undo failed because a log is corrupt.

The log record that caused the error is identified as follows:

type	Is the logical-log record type.
subtype	Is the logging subsystem.
nn	Is the logical-log ID where the record is stored.
0xn	Is the hexadecimal address position within the log.

Action: To determine if any action is needed, use the onlog utility to examine the logical log. Contact Technical Support.

Log record (*type:subtype***) failed, partnum** *pnum row rid* **iserrno** *num.*

Cause: A logging failure occurred.

The log record that caused the error is identified as follows:

type	Is the logical-log record type.
subtype	Is the logging subsystem.
pnum	Is the part number.
rid	Is the row ID.
num	Is the iserror number.

Action: Contact Technical Support.

Log record (*type:subtype***) in log** *nn***, offset** *0xn* **was not rolled back.**

Cause: A log undo failed because a log is corrupt.

The log record that caused the error is identified as follows:

type	Is the logical-log record type.
subtype	Is the logging subsystem.
log	Is the logical-log ID where the record is stored.
offset	Is the hexadecimal address position within the log.

Action: To determine if any action is needed, use the onlog utility to examine the logical log. Contact Technical Support.

Logical Recovery allocating *nn* **worker threads** *thread_type***.**

Cause: The database server determined the number of worker threads that will be used for parallel recovery. The variable *thread_type* can assume the values ON_RECVRY_THREADS or OFF_RECVRY_THREADS.

Action: This status message requires no action. If you want a different number of worker threads allocated for parallel recovery, change the value of the ONCONFIG configuration parameter ON_RECVRY_THREADS or OFF_RECVRY_THREADS.

Logical Recovery Started.

Cause: Logical recovery began.

Action: This status message requires no action.

Maximum server connections *number***.**

Cause: Outputs with each checkpoint message to indicate the maximum number of concurrent connections to the database server since the last restart.

Action: This message helps the customer track license usage to determine when more licenses need to be purchased. For assistance, Contact Technical Support.

Memory allocation error.

Cause: The database server ran out of shared memory.

Action: Take one of the following actions:

1. Increase swap space on the computer.
2. Check kernel shared-memory parameters for limits on shared memory.
3. Decrease the size of the memory allocated, with the **buffers** field in the BUFFERPOOL configuration parameter.
4. Increase the virtual-memory size (SHMVIRTSIZE), the size of the added segments, (SHMADD), or your total shared-memory size (SHMTOTAL).

Mirror Chunk *chunkname* **added to space** *spacename***. Perform manual recovery.**

Cause: Fast recovery, full recovery, or an HDR secondary has recovered the add of a mirror chunk. It does not perform automatic mirror recovery, however. The administrator must do this.

Action: Use either the **onspaces** utility or ON–Monitor to attempt to recover the mirror chunks.

Mixed transaction result. (*pid*=*nn* **user=***userid***).**

Cause: You receive this message only when more than one database server is involved in a transaction. This message indicates that a database server, after preparing a transaction for commit, heuristically rolled back the transaction, and the global transaction completed inconsistently. The *pid* value is the user-process identification number of the coordinator process. The value of *user* is the user ID associated with the coordinator process.

Action: See the information on recovering manually from failed two-phase commit in your *IBM Informix Administrator's Guide*.

mt_shm_free_pool: pool *0xn* **has blocks still used (id** *nn***).**

Cause: An internal error occurred during a pool deallocation because blocks are still associated with the pool.

Action: Contact Technical Support.

mt_shm_init: can't create *resident/virtual* segment.

Cause: The causes for the failure to create the resident or virtual segment are as follows: (1) the segment size is less than the minimum segment size; (2) the segment size is larger than the maximum segment size; (3) allocating another segment would exceed the allowable total shared-memory size; or (4) a failure occurred while the database server was trying to allocate the segment.

Action: If you suspect that this error was generated because of item 1 or 2 in the preceding paragraph, Contact Technical Support. To correct item 3, increase the SHMTOTAL value in your ONCONFIG configuration file. For additional information about errors generated because of item 4, see your logical-log file.

mt_shm_remove: WARNING: may not have removed all/correct segments.

Cause: When the operating system tried to remove the shared-memory segments associated with the database server, the last segment did not equal the last segment registered internally. This situation is probably due to the unexpected failure of the database server.

Action: Remove any segments that were not cleaned up.

Messages: N-O-P

Newly specified value of *value* for the pagesize in the configuration file does not match older value of *value*. Using the older value.

Cause: This message displays upon database server restart. The PAGESIZE value changed in the ONCONFIG file after the database server was initialized.

Action: The database server uses the older PAGESIZE value.

Not enough main memory.

Cause: The database server detected an error in an attempt to acquire more memory space from the operating system.

Action: For more information about shared-memory configuration and management, refer to your operating-system documentation.

Not enough logical-log files, Increase LOGFILES.

Cause: During a data restore, the value of the LOGFILES configuration must always be greater than or equal to the total number of logical-log files. At some point during the restore, the number of logical-log files exceeded the value of LOGFILES.

Action: Increase the value of LOGFILES in ONCONFIG.

Not enough physical procs for affinity.

Cause: The ONCONFIG parameters AFF_NPROCS and AFF_SPROC are not correctly set. AFF_SPROC plus AFF_NPROCS is greater than the number of physical processors on your computer or node.

Action: Reset AFF_NPROCS and AFF_SPROC, such that the value AFF_SPROC plus value of AFF_NPROCS is less than or equal to the number of physical processors.

The number of configured CPU poll threads exceeds NUMCPUVPS.

Cause: The number of in-line poll threads that you specified in the ONCONFIG configuration file exceeds the number of CPU virtual processors.

Action: Reduce the number of in-line poll threads to be less than or equal to the number of CPU virtual processors.

onconfig parameter *parameter* **modified from** *old_value* **to** *new_value.*

Cause: When the database server shared memory is reinitialized, this message documents any changes that occurred since the last initialization.

Action: None required.

oninit: Cannot have SINGLE_CPU_VP non-zero and number of CPU VPs greater than 1.

Cause: The ONCONFIG file contains VPCLASS cpu with a `num=` value greater than 1 and a nonzero value for SINGLE_CPU_VP. SINGLE_CPU_VP must be 0 (or omitted) when there are more than 1 CPU VPs.

Action: Correct the ONCONFIG file and restart the database server.

oninit: Cannot have SINGLE_CPU_VP non-zero and user-defined VP classes.

Cause: The ONCONFIG file contains a user-defined VPCLASS as well as a nonzero value for SINGLE_CPU_VP. SINGLE_CPU_VP must be `0` (or omitted) when the ONCONFIG file contains a user-defined VPCLASS.

Action: Correct the ONCONFIG file and restart the database server.

oninit: Cannot mix VPCLASS cpu and NUMCPUVPS,

AFF_SPROC, AFF_NPROCS, or NOAGE parameters.

Cause: The ONCONFIG file contains both VPCLASS cpu and one or more of the other listed parameters. It cannot contain both.

Action: Correct the ONCONFIG file and restart the database server.

---

oninit: Cannot mix VPCLASS aio and NUMAIOVPS parameters.

Cause: The ONCONFIG file contains both VPCLASS aio and NUMAIOVPS. It cannot contain both.

Action: Correct the ONCONFIG file and restart the database server.

---

oninit: Fatal error in initializing ASF with 'ASF_INIT_DATA' flags asfcode = '25507'.

Cause: The **nettype** value specified in the **sqlhosts** file or registry for the database server is invalid or unsupported, or the **servicename** specified in the **sqlhosts** file or registry for the database server is invalid.

Action: Check the **nettype** and **servicename** values in the **sqlhosts** file or registry for each DBSERVERNAME and for the DBSERVERALIASES. Check the **nettype** value in each NETTYPE parameter in the ONCONFIG file.

---

oninit: invalid or missing name for Subsystem Staging Blobspace.

Cause: You set the configuration parameter STAGEBLOB to a blobspace that does not exist.

Action: Use the **-d** option of **onspaces** to create the blobspace specified in STAGEBLOB, and restart the database server.

---

oninit: Too many VPCLASS parameters specified.

Cause: Too many VPCLASS parameter lines have been specified in the ONCONFIG file.

Action: Reduce the number of VPCLASS lines, if possible. If not possible, contact Technical Support.

---

oninit: VPCLASS *classname* **bad affinity specification.**

Cause: The affinity specification for the VPCLASS line is incorrect. Affinity is specified as a range:

For *m*, use processor *m*.

For *m* to *n*, use processors in the range *m* to *n* inclusive, where *m* `<=`*n*, *m* `>= 0`, and *n* `>= 0`.

Action: Correct the VPCLASS parameter in the ONCONFIG file and restart the database server.

---

oninit: VPCLASS *classname* **duplicate class** *name***.**

Cause: The VPCLASS *classname* in the ONCONFIG file has a duplicate name. VP class names must be unique.

Action: Correct the duplicate name and restart the database server.

---

oninit: VPCLASS *classname* **illegal option.**

Cause: One of the fields in the VPCLASS *classname* parameter is illegal.

Action: Correct the parameter in the ONCONFIG file and restart the database server.

---

oninit: VPCLASS *classname* **maximum number of VPs is out of the range 0-10000.**

Cause: The maximum number of VPs specified by a VPCLASS parameter line must be in the range 1 to 10,000.

Action: Correct the value and restart the database server.

---

oninit: VPCLASS *classname* **name is too long. Maximum length is** *maxlength***.**

Cause: The length of the name field in VPCLASS *classname* is too long.

Action: Choose a shorter class name, correct the ONCONFIG file, and restart the database server.

---

oninit: VPCLASS *classname* **number of VPs is greater than the maximum specified.**

Cause: The initial number of VPs specified by a VPCLASS parameter is greater than the maximum specified by the same VPCLASS parameter.

Action: Correct the VPCLASS parameter and restart the database server.

---

oninit: VPCLASS *classname* **number of VPs is out of the range 0-10000.**

Cause: The initial number of VPs specified by a VPCLASS parameter line must be in the range 1 to 10,000.

Action: Correct the value and restart the database server.

---

onmode: VPCLASS *classname* **name is too long. Maximum length is** *maxlength***.**

Cause: The name of a dynamically added VP class that **onmode -p** specifies is too long.

Action: Choose a shorter name, and retry the **onmode -p** command.

Optical Subsystem is running.

Cause: You set the value of the STAGEBLOB parameter in the configuration file, and the database server is communicating properly with the optical-storage subsystem.

Action: No action is required.

Optical Subsystem is not running.

Cause: You set the value of the STAGEBLOB parameter in the configuration file, but the database server cannot detect the existence of the optical-storage subsystem.

Action: Check that the optical subsystem is online.

Optical Subsystem STARTUP Error.

Cause: The database server detects that the optical-storage subsystem is running, but the database server cannot communicate with it properly.

Action: Check your optical subsystem for errors.

Online Mode.

Cause: The database server is in online mode. Users can access all databases

Action: This status message requires no action.

onspaces: unable to reset dataskip.

Cause: This error message comes from the **onspaces** utility. For some reason, the utility cannot change the specification of DATASKIP (ON or OFF) across all dbspaces in the database server instance.

Action: You are unlikely to receive this message. If the error persists after you restart the database server, Contact Technical Support.

Open transaction detected when changing log versions.

Cause: The database server detected an open transaction while it was trying to convert the data from a previous version of the database server.

Action: Conversion is not allowed unless the last record in the log is a checkpoint. You must restore the previous version of the database server, force a checkpoint, and then retry conversion.

Out of message shared memory.

Cause: The database server could not allocate more memory for the specified segment.

Action: For additional information, see the log file.

Out of resident shared memory.

Cause: The database server could not allocate more memory for the specified segment.

Action: For additional information, see the log file.

Out of virtual shared memory.

Cause: The database server could not allocate more memory for the specified segment.

Action: For additional information, see the log file.

PANIC: Attempting to bring system down.

Cause: A fatal database server error occurred.

Action: See the error that caused the panic and attempt the corrective action suggested by the error message. For additional information that might explain the failure, refer also to other messages in the message-log file.

Participant site *database_server* **heuristically rolled back.**

Cause: A remote site rolled back a transaction after it reached the prepared-for-commit phase.

Action: You might need to roll back the transaction on other sites and then restart it.

Physical recovery complete: *number* **pages examined,** *number* **pages restored.**

Cause: This message displays during fast recovery. The *number of pages examined* indicates the number of page images that exist in the physical log. The *number of pages restored* indicates the actual number of pages that are restored from the physical log. The number of pages restored is always less than or equal to the number examined.

The database server might physically log a page image multiple times between checkpoints. Physical recovery restores only the first logged page image.

If a page stays in the memory buffer pool, the database server physically logs it once per checkpoint, and stores one page image in the physical log. If the buffer pool is too small, a page that is being updated many times might get forced out of the buffer pool to disk and then brought back into memory for the next update. Each time the page is brought into memory, it is physically logged again, resulting in duplicate page images in the physical log.

Action: If the *number of pages examined* is much larger than the *number of pages restored,* increase the size of the buffer pool to reduce the number of duplicate before-images. For more information, see the *IBM Informix Performance Guide*.

Physical recovery started at page (*chunk*:*offset*).

Cause: This message displays during fast recovery. *Chunk* is the number of the chunk that contains the physical log. *Offset* is the page offset of the start of the physical log entries. Physical recovery begins restoring pages from that point.

Action: No action required. For information on fast recovery, see the *IBM Informix Administrator's Guide*.

Portions of partition partnum of table tablename in database dbname were not logged. This partition cannot be rolled forward.

Cause: Light appends occurred to the operational table since the last backup.

Action: If you want full access to data in this table, you need to alter the table to raw and then to the desired table type. This alter operation removes inconsistencies in the table that resulted from replaying non-logged operations such as light appends.

Possible mixed transaction result.

Cause: This message indicates that error -716 has been returned. Associated with this message is a list of the database servers where the result of a transaction is unknown.

Action: For information on determining if a transaction was implemented inconsistently, see the *IBM Informix Administrator's Guide*.

Prepared participant site *server_name* **did not respond.**

Cause: Too many attempts were made to contact remote site *server_name*. After several timeout intervals were met, the site was determined to be down.

Action: Verify that the remote site is online and that it is correctly configured for distributed transactions. Once the remote site is ready, reinitiate the transaction.

Prepared participant site *server_name* **not responding.**

Cause: The database server is attempting to contact remote site *server_name*. For some unknown reason, the database server cannot contact the remote site.

Action: Verify that the remote site is online and that it is correctly configured for distributed transactions.

Messages: Q-R-S

Quiescent Mode.

Cause: The database server has entered quiescent mode from some other state. On UNIX, only users logged in as **informix** or as **root** can interact with the database server. On Windows, only members of the **Informix-Admin** group can interact with the database server. No user can access a database.

Action: None required.

Read failed. Table *name*, **Database** *name*, **iserrno =** *number*

Cause: An error occurred reading the specified system table.

Action: Note the error number and contact Technical Support.

Recovery Mode.

Cause: The database server entered the recovery mode. No user can access a database until recovery is complete.

Action: None required.

Recreating index: '*dbsname*:"*owner*".*tabname-idxname***'.**

Cause: This message indicates which index is currently being re-created.

Action: None required.

Rollforward of log record failed, iserrno = *nn*.

Cause: The message appears if, during fast recovery or a data restore, the database server cannot roll forward a specific logical-log record. The database server might be able to change to quiescent or online mode, but some inconsistency could result. For further information, see the message that immediately precedes this one. The *iserrno* value is the error number.

Action: Contact IBM Informix Technical Support.

Root chunk is full and no additional pages could be allocated to chunk descriptor page.

Cause: The root chunk is full.

Action: To free space in the root chunk, take one of the following actions:
- Drop and re-create the **sysmaster** database.
- Move user tables from the root dbspace to another dbspace.
- Refragment tables.

scan_logundo: subsys *ss*, **type** *tt*, **iserrno** *ee*.

Cause: A log undo failed because log type *tt* is corrupt.

The variables in this message have the following values:

ss Is the subsystem name.

tt Is the logical-log record type.

ee Is the iserror number.

Action: Examine the logical log with the onlog utility to determine if any action is needed. Contact Technical Support.

Session completed abnormally. Committing *tx* **id** *0xm***, flags** *0xn***.**

Cause: Abnormal session completion occurs only when the database server is attempting to commit a transaction that has no current owner, and the transaction develops into a long transaction. The database server forked a thread to complete the commit.

Action: None required.

Session completed abnormally. Rolling back *tx* **id** *0xm***, flags** *0xn.*

Cause: Abnormal session completion occurs only when the database server is attempting to commit a distributed transaction that has no current owner, and the transaction develops into a long transaction. The database server forked a thread that rolled back the transaction.

Action: None required.

---

semctl: errno = *nn.*

Cause: When the database server initialized a semaphore, an error occurred. The operating-system error is returned.

Action: See your operating-system documentation.

---

semget: errno = *nn.*

Cause: An allocation of a semaphore set failed. The operating-system error is returned.

Action: See your operating-system documentation.

---

shmat: *some_string os_errno***:** *os_err_text***.**

Cause: An attempt to attach to a shared-memory segment failed. The system error number and the suggested corrective action are returned.

Action: Review the corrective action (if given), and determine if it is reasonable to try. For more information, refer to your operating-system documentation.

---

shmctl: errno = *nn.*

Cause: An error occurred while the database server tried to remove or lock a shared-memory segment. The operating-system error number is returned.

Action: See your operating-system documentation.

---

shmdt: errno = *nn.*

Cause: An error occurred while the database server was trying to detach from a shared-memory segment. The operating-system error number is returned.

Action: See your operating-system documentation.

---

shmem sent to *filename.*

Cause: The database server wrote a copy of shared memory to the specified file as a consequence of an assertion failure.

Action: None.

---

shmget: *some_str os_errno***: key** *shmkey***:** *some_string.*

Cause: Either the creation of a shared-memory segment failed, or an attempt to get the shared-memory ID associated with a certain key failed. The system error number and the suggested corrective action are returned.

Action: Consult your operating-system documentation.

Shutdown (onmode -k) or override (onmode -O).

Cause: A dbspace has gone down during a checkpoint interval. The database server is configured to wait for an override when this situation occurs.

When the checkpoint actually happens, the following message appears: `Checkpoint blocked by down space, waiting for override or shutdown`.

Action: Either shut down the database server or issue an **onmode -O** command to override the down dbspace. For more information on the **onmode** utility, see "In This Chapter" on page 10-1.

Shutdown Mode.

Cause: The database server is in the process of moving from online mode to quiescent mode.

Action: None required.

Space *spacename* **added**.

Cause: The database server administrator added a new storage space *spacename* to the database server.

Action: None required.

Space *spacename* **dropped**.

Cause: The database server administrator dropped a storage space *spacename* from the database server.

Action: None required.

Space *spacename* **-- Recovery Begins(***addr***).**

Cause: This informational message indicates that the database server is attempting to recover the storage space.

The variables in this message have the following values:

spacename
: Is the name of the storage space that the database server is recovering.

addr
: Is the address of the control block.

Action: None required.

Space *spacename* **-- Recovery Complete(***addr***).**

Cause: This informational message indicates that the database server recovered the storage space.

The variables in this message have the following values:

spacename
: Is the name of the storage space that the database server has recovered.

addr
: Is the address of the control block.

Action: None required.

Space *spacename* **-- Recovery Failed(***addr***).**

Cause: This informational message indicates that the database server was unable to recover the storage space.

The variables in this message have the following values:

spacename
: Is the name of the storage space that the database server failed to recover.

addr
: Is the address of the control block.

Action: None required.

sysmaster database built successfully.

Cause: The database server successfully built the sysmaster database.

Action: None required.

Successfully extend physical log space

Cause: The physical log space was successfully extended to the file plog_extend.*servernum* under the designated path.

Action: None required.

Messages: T-U-V

This ddl operation is not allowed due to deferred constraints pending on this table and dependent tables.

Cause: This error is returned when you attempt to start a violations table and constraints are in deferred mode.

Note: No error is returned if you start a violations table and then later set the constraints to deferred. However, the violations get undone immediately rather than written into the deferred constraint buffer. For more information, see the *IBM Informix Guide to SQL: Syntax*.

Action: If you would like to start a violations table, you must either change the constraint mode to immediate or commit the transaction.

This type of space does not accept log files.

Cause: Adding a logical-log file to a blobspace or sbspace is not allowed.

Action: Add the logical-log file to a dbspace. For more information, see "Add a Logical-Log File" on page 12-2.

TIMER VP: Could not redirect I/O in initialization, errno = *nn.*

Cause: The operating system could not open the null device or duplicate the file descriptor associated with the opening of that device. The system error number is returned.

Action: See your operating-system documentation.

Too Many Active Transactions.

Cause: During a data restore, there were too many active transactions. At some point during the restore, the number of active transactions exceeded 32 kilobytes.

Action: None.

Too many violations.

Cause: The number of violations in the diagnostics table exceeds the limit that is specified in the MAX VIOLATIONS clause of the START VIOLATIONS TABLE statement. When a single statement on the target table (such as an INSERT or UPDATE statement) inserts more records into the violations table than the limit that is specified by the MAX VIOLATIONS clause, this error is returned to the user who issued the statement on the target table.

This MAX VIOLATIONS limit applies to each coserver. For example, if you reach the MAX VIOLATIONS limit on coserver 2, you can continue to issue statements that violate rows on other coservers until you reach the MAX VIOLATIONS limit.

Action: To resolve this error, perform one of the following actions:

- Omit the MAX VIOLATIONS clause in the START VIOLATIONS TABLE statement when you start a violations table. Here, you are specifying no limit to the number of rows in the violations table.
- Set MAX VIOLATIONS to a high value.

Transaction Not Found.

Cause: The logical log is corrupt. This situation can occur when a new transaction is started, but the first logical-log record for the transaction is not a BEGWORK record.

Action: Contact Technical Support.

Transaction heuristically rolled back.

Cause: A heuristic decision occurred to roll back a transaction after it completed the first phase of a two-phase commit.

Action: None required.

Transaction table overflow - user id *nn*, **process id** *nn.*

Cause: A thread attempted to allocate an entry in the transaction table when no entries in the shared-memory table were available. The user ID and process ID of the requesting thread are displayed.

Action: Try again later.

Unable to create output file *filename* **errno =** *nn.*

Cause: The operating system cannot create output file *filename*. The *errno* is the number of the operating-system error returned.

Action: Verify that the directory exists and has write permissions.

Unable to extend *nn* **reserved pages for** *purpose* **in root chunk.**

Cause: The operating system cannot extend to *nn* reserved pages for *purpose* in root chunk. (The value *purpose* can be either Checkpoint/Log, DBSpace, Chunk, or Mirror Chunk.)

Action: Reduce the ONCONFIG parameter for the resource cited; bring the database server up and free some space in the primary root chunk. Then reattempt the same operation.

Unable to initiate communications with the Optical Subsystem.

Cause: The optical driver supplied by the optical-drive vendor has indicated that the drive is not accessible.

Action: Check driver installation and cabling between the computer and the drive.

Unable to start SQL engine.

Cause: The database server encountered an out-of-memory condition.

Action: No action is necessary.

Unable to open tblspace *nn*, iserrno = *nn*.

Cause: The database server cannot open the specified tblspace. (The value *nn* is the hexadecimal representation of the tblspace number.)

Action: See the ISAM error message number *nn*, which should explain why the tblspace cannot be accessed. The error message appears in *IBM Informix Error Messages* at the IBM Informix Online Documentation site at: www.ibm.com/software/data/developer/informix.

The value of pagesize *pagesize* specified in the config file is not a valid pagesize. Use 2048, 4096 or 8192 as the value for PAGESIZE in the onconfig file and restart the server.

Cause: This message displays upon disk initialization. The value of PAGESIZE that was specified in the ONCONFIG file is not a valid value.

Action: Restart the database server with a valid PAGESIZE value.

Violations table is not started for the target table.

Cause: If you issue a STOP VIOLATIONS TABLE statement for which no violations table is started, you receive this message.

Action: To recover from this error, you must start a violations table for the target table.

Violations table reversion test completed successfully.

Cause: This message is recorded in the **logmessage** table in the **sysmaster** database when the **revtestviolations.sh** script has completed successfully (no open violations tables were found).

Action: No action is necessary. For more information on **revtestviolations.sh**, see the *IBM Informix Migration Guide*.

Violations table reversion test failed.

Cause: When the database server finds an open violations table, it reports errors 16992 and 16993 in the **logmessage** table in the **sysmaster** database and aborts the reversion process.

Action: When this message appears, you must issue the STOP VIOLATIONS TABLE FOR *table_name* command for each open violations table. After you close all open violations tables, you can restart the reversion process.

Violations table reversion test start.

Cause: This message is recorded in the **logmessage** table in the **sysmaster** database when the **revtestviolations.sh** script is executed.

Action: No action is necessary. For more information on **revtestviolations.sh**, see the *IBM Informix Migration Guide*.

Violations tables still exist.

Cause: This message is recorded in the **logmessage** table in the **sysmaster** database when an open violations table is found.

Action: When this message appears, you must issue the STOP VIOLATIONS TABLE FOR *table_name* command for each open violations table. After you close all open violations tables, you can restart the reversion process.

Virtual processor limit exceeded.

Cause: You configured the database server with more than the maximum number of virtual processors allowed (1000).

Action: To reduce the number of virtual processors, decrease the values of VPCLASS, NUMCPUVPS, NUMAIOVPS, or NETTYPE in your ONCONFIG file.

VPCLASS *classname* name is too long. Maximum length is *maxlength*.

Cause: This message indicates an internal error.

Action: Contact Technical Support.

VPCLASS *classname* duplicate class name.

Cause: This message indicates an internal error.

Action: Contact Technical Support.

VPCLASS *classname* Not enough physical procs for affinity.

Cause: The physical processors in the affinity specification for the VP class *classname* do not exist or are offline. The problem might be with the VPCLASS parameter for cpu class VPs or with the AFF_SPROC and AFF_NPROCS parameters.

Action: Make sure the named processors are online. Correct the affinity specification for the named VP class. Restart the database server.

Messages: W-X-Y-Z

WARNING: aio_wait: errno = *nn.*

Cause: While the database server was waiting for an I/O request to complete, it generated error number *nn* on an operation that it was attempting to execute.

Action: Contact Technical Support for assistance.

WARNING: Buffer pool size may cause database server to get into a locked state. Recommended minimum buffer pool size is *num* **times maximum concurrent user threads.**

Cause: There are not enough buffers in the buffer pool. The database server could use all available buffers and cause a deadlock to occur.

Action: Change the **buffers** field in the BUFFERPOOL parameter in the ONCONFIG file to the number that this message recommends. For more information on the BUFFERPOOL parameter, see "BUFFERPOOL" on page 1-14..

warning: Chunk time stamps are invalid.

Cause: A sanity check is performed on chunks when they are first opened at system initialization. The chunk specified did not pass the check and will be brought offline.

Action: Restore the chunk from a dbspace backup or its mirror.

Warning: *name_old* **is a deprecated onconfig parameter. Use** *name_new* **instead. See the release notes and the Informix Administrator's Reference for more information.**

Cause: A deprecated ONCONFIG parameter was used. This message displays the first time that you use a deprecated parameter. The shorter form of the message displays thereafter.

Action: Use the suggested alternative ONCONFIG parameter.

Warning: *name_old* **is a deprecated onconfig parameter. Use** *name_new* **instead.**

Cause: A deprecated ONCONFIG parameter was used.

Action: Use the suggested alternative ONCONFIG parameter.

Warning: Unable to allocate requested big buffer of size *nn.*

Cause: The internal memory allocation for a big buffer failed.

Action: Increase either virtual memory size (SHMVIRTSIZE), the size of the added segments (SHMADD), or your total shared-memory size (SHMTOTAL).

You are turning off smart large object logging.

Cause: These changes will become the new sbspace default values. Changes have been made to the sbspace. The onspaces utility will read and update 100 smart large objects at a time and commit each block of 100 smart large objects as a single transaction. This utility might take a long time to complete.

Action: This informational message occurs when you issue the following command:

```
onspaces -ch sbspace -Df "LOGGING=OFF" -y
```

For more information, see "Change Sbspace Default Specifications" on page 13-14.

Messages: Symbols

HH:MM:SS **Informix database server Version** *R.VV.PPPPP* **Software Serial Number RDS#***XYYYYYY***.**

Cause: This message indicate the start-up of the database server, after the initialization of shared memory.

Action: No action is required.

*argument***: invalid argument.**

Cause: This internal error indicates that an invalid argument was passed to an internal routine.

Action: Contact Technical Support.

*function_name***: cannot allocate memory.**

Cause: The database server cannot allocate memory from internal shared-memory pool.

Action: Increase either virtual-memory size (SHMVIRTSIZE), the size of the added segments (SHMADD), or your total shared-memory size (SHMTOTAL).

Conversion/Reversion Messages

These messages might display during database server conversion or reversion.

Messages: A-C

Cannot revert constraint with id *id* (in syschecks).

Cause: The database has a constraint that was defined in a version more recent than the one to which you are reverting.

Action: Drop the specified constraint and retry reversion.

Cannot revert new fragment expression for index *index*, tabid *id*.

Cause: The index fragmentation was defined in a version more recent than the one to which you are reverting.

Action: Drop the problem index-fragmentation scheme and retry reversion.

Cannot revert new table fragment expression for *table* with id *id*.

Cause: The fragmentation of this table was defined in a version more recent than the one to which you are reverting.

Action: Drop the problem table fragmentation scheme and retry reversion.

Cannot update page zero.

Cause: Attempt to write page zero failed.

Action: Contact Technical Support.

Checking database *name* for revertibility.

Cause: Indicates that start of the reversion checks on the specified database.

Action: None required.

Conversion of pre 7.3 in-place alter started *status*.

Cause: The database server is converting data structures for in-place alters to the new format.

Action: None required.

Conversion of pre 9.2 database tablespaces *status*.

Cause: The database server is converting tablespaces to the new format.

Action: None required.

The conversion of the database *name* has failed.

Cause: Indicates that the conversion of the specified database has failed.

Action: Connect to the database. This action triggers conversion of the database. If it fails, the relevant error message appears. Contact Technical Support.

Converting database *name*...

Cause: This message appears at the start of conversion of each database in the system.

Action: None required.

Converting in-place alters to new format.

Cause: The database server is converting data structures for in-place alters to the new format.

Action: None required.

Converting 'onpload' database...

Cause: Printed in **online.log** at the beginning of **onpload** conversion.

Action: None required.

Converting partition header from version 7.x.

Cause: The database server is converting the partition header page to the new format that contains the chunk number and offset.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action: None required.

Converting partition header page *address*.

Cause: The database server is converting the partition header page to the new format that contains the chunk number and page offset.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action: None required.

Converting partition header pages *status*.

Cause: This message tracks the progress of the conversion of the partition header pages. The status is identified as follows:

- `started`
- `succeeded`

- `FAILED`

Action: If the status is started or succeeded, no action is required.

If conversion of the partition header pages failed, restart the database server. It will attempt to continue converting where it left off in the restartable conversion phase. If this action fails, diagnose the problem, restore from tape, fix the problem, and retry conversion.

Converting partition keys to 9.2.

Cause: The database server is converting the partition keys to the Version 9.2 format.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action: None required.

Converting partition name for *databasename*:*tablename***.**

Cause: The database server is converting the partition name for the *databasename*:*tablename*.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action: None required.

Messages: D-F

The database *name* **has been converted successfully.**

Cause: Indicates successful completion of the conversion of the specified database.

Action: None required.

Database *name* **is not revertible...**

Cause: The database has failed one of the reversion checks and is not revertible.

Action: Take action to correct the error displayed as a separate message.

Database *name* **is revertible...**

Cause: The database has passed all reversion checks and is revertible to the specified version.

Action: None required.

Database *name***: Must drop trigger (id =** *id_number***).**

Cause: The database contains a trigger that was created in a version more recent than the one to which you are converting.

Action: Drop the trigger with the specified trigger identification number and then attempt reversion.

Database *name* **SUCCESSFULLY reverted...**

Cause: Indicates the success of reversion of the specified database.

Action: None required.

... dropping sysmaster database.

Cause: The database server is dropping sysmaster database during the reversion process.

Action: No action is required.

The dummy updates failed while converting database *name***. This may imply data corruption in the database. If so, restore the original database with the tape backup. For more information, see** *output_file***.**

Cause: During conversion of a database from a version earlier than Version 9.2, dummy update statements are run against the system tables in the database being converted. This message indicates failure in running one of these update statements.

Action: To retry the dummy updates, run the dummy update script for your old database server version. For instructions, refer to the *IBM Informix Migration Guide*.

If data corruption occurred, restore the original database with the tape backup. For more information, see the *IBM Informix Backup and Restore Guide*.

The dummy updates succeeded while converting database *name***.**

Cause: During conversion of a database from a version earlier than Version 9.2, dummy update statements are run against the system tables in the database being converted. This message indicates successful completion of these updates.

Action: None required.

Error in slow altering a system table.

Cause: An internal error occurred while performing reversion.

Action: Contact Technical Support.

External conversion aborted due to incompatible sysmaster database.

Cause: The **sysmaster** database was not converted to the current database server version. A current **sysmaster** database is needed for external conversion to complete.

Action: Drop the **sysmaster** database and reboot the database server. It will build a new **sysmaster** database and relaunch external conversion automatically.

Messages: I-P

Internal server error.

Cause: An unexpected error occurred during database reversion.

Action: Contact Technical Support.

Must drop long identifiers in table *name* **in database** *name*

Cause: Identifiers greater than 18 characters in length are not supported in the database server version to which you are reverting.

Action: Make sure that all long identifiers in the system are either dropped or renamed before you attempt reversion.

Must drop new database (*name***) before attempting reversion. Iserrno** *error_number*

Cause: The system contains a database that was created in a more recent version of the database server.

Action: Drop the new database and attempt reversion.

Must drop new user defined statistics in database *name***, iserrno** *number*

Cause: Some distributions in the **sysdistrib** system table use user-defined statistics. This feature is not supported in the version to which you are reverting.

Action: Ensure that no user-defined statistics are present or used in the system and then attempt reversion.

ON–Bar conversion completed successfully.

Cause: ON–Bar conversion completed successfully.

Action: None.

ON–Bar conversion failed see /tmp/bar_conv.out.

Cause: ON–Bar conversion failed.

Action: For failure details, see **/tmp/bar_conv.out**.

ON–Bar conversion start:

Cause: ON–Bar conversion script is now running.

Action: None.

ON–Bar reversion completed successfully.

Cause: ON–Bar reversion was completed successfully.

Action: None.

ON–Bar reversion failed see /tmp/bar_rev.out.

Cause: ON–Bar reversion failed.

Action: For failure details, see **/tmp/bar_rev.out**.

ON–Bar reversion start:

Cause: ON–Bar reversion script is now running.

Action: None.

ON–Bar reversion test completed successfully.

Cause: ON–Bar reversion test was completed successfully.

Action: None.

ON–Bar reversion test start:

Cause: ON–Bar reversion test script is now running.

Action: None.

'onpload' conversion completed successfully.

Cause: Displayed in **online.log** at the successful completion of **onpload** conversion.

Action: None required.

'onpload' conversion failed. For details, look in $INFORMIXDIR/etc/conpload.out.

Cause: Conversion of the **onpload** database failed.

Action: Find out the cause of failure from $INFORMIXDIR**/etc/conpload.out**. Fix the problem before you reattempt conversion.

...'onpload' reversion completed successfully.

Cause: Printed in **online.log** at the successful completion of reversion.

Action: None required.

...'onpload' reversion failed. For details, look in $INFORMIXDIR/etc/revpload.out.

Cause: Reversion of the **onpload** database failed.

Action: Find the cause of failure in **$INFORMIXDIR/etc/revpload.out**. Fix the problem before you reattempt reversion.

'onpload' reversion test completed successfully.

Cause: Printed in **online.log** if the **onpload** database is revertible.

Action: None required.

'onpload' reversion test start:

Cause: Printed in **online.log** at the beginning of **onpload** reversion testing.

Action: None required.

The pload database contains load/unload jobs referring to long table names, column names, or database names. These jobs will not work as expected until they are redefined.

Cause: Printed during **onpload** reversion testing if the **onpload** database contains references to long table names, column names, or database names. But the reversion will complete.

Action: Redefine the load and unload jobs in the **onpload** database that have references to long identifiers.

Messages: R-W

...reverting 'onpload' database.

Cause: Printed in **online.log** at the beginning of **onpload** reversion.

Action: None required.

Reverting partition header from version 9.2.

Cause: The database server is reverting the partition header page to the old format that contains the physical address.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action: None required.

Reverting partition header page *address***.**

Cause: The database server is reverting the partition header page to the old format that contains the physical address.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action: None required.

Reverting partition header pages *status***.**

Cause: The database server is reverting the partition header pages to the old format. The status is identified as follows:

- `started`
- `succeeded`
- `FAILED`

Action: If reversion of the partition header pages started or succeeded, no action is required. If reversion of the partition header pages failed, restore from a tape backup, diagnose and fix the problem, and retry conversion.

Reverting partition keys to pre 9.2.

Cause: The database server is reverting the partition keys to the pre-Version 9.2 format.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action: None required.

Reverting partition *name* **for** *databasename***:***tablename***.**

Cause: The database server is reverting the partition name for *databasename:tablename*.

This message is optional verbose output that is logged only if you start **oninit** with the **-v** flag.

Action: None required.

... reverting reserved pages.

Cause: The database server is reverting reserved pages.

Action: No action is required.

... reverting tables that underwent In-Place Alter.

Cause: The database server is reverting tables that underwent in-place alter.

Action: No action is required.

R-tree error message conversion completed successfully.

Cause: R-tree error message conversion was completed successfully.

Action: None required

R-tree error message conversion failed. (See /tmp/conrtree.out or %TMP%\conrtree.out)

Cause: R-tree error message conversion failed.

Action: See /**tmp/conR-tree.out** and **/tmp/R-tree.databases**.

R-tree error message conversion started.

Cause: R-tree error message conversion script is now running.

Action: None required.

Reversion cancelled.

Cause: The reversion process was cancelled because of errors encountered.

Action: Correct the cause of the errors, and restart reversion.

Reversion complete. Install IBM Informix database server *version* **before restarting.**

Cause: The reversion process was completed successfully.

Action: You must install the older database version.

Reversion of database *name* **FAILED**

Cause: Indicates the failure of reversion of the specified database.

Action: None required.

...reverting 'syscdr' database.

Cause: Printed in **online.log** at the beginning of Enterprise Replication reversion.

Action: None required.

...starting reversion of database *name***.**

Cause: Indicates the start of actual reversion of the specified database.

Action: None required.

There is a semi-detached index in this table, which cannot be reverted. Drop this index, and retry reversion.

Cause: A semi-detached index on this table cannot be reverted.

Action: To see the list of all semi-detached indexes, refer to the database server message log. These indexes cannot be reverted. To continue reversion, drop these semi-detached indexes and retry reversion. If needed, you will need to re-create these indexes after reversion is complete.

Unable to read reserved page *chunk:offset* **-** *reserved_page***.**

Cause: Both disk pages in a given reserved page pair are bad. On the disk page, c*hunk* represents the chunk number and *offset* represents the page offset for the chunk.

Action: Contact Technical Support.

WARNING: Target server version must have a certified Storage Manager installed after conversion/reversion and before bringing up server.

Cause: ON–Bar is being converted or reverted. The user must ensure that a storage manager, certified with the target database server version, is installed.

Action: None.

Conversion and Reversion Messages for Enterprise Replication

Use the **concdr.sh** script on UNIX or the **concdr.bat** script on Windows to convert Enterprise Replication and the **syscdr** database to Version 10.0. Use the **revcdr.sh** script on UNIX the or **revcdr.bat** script on Windows to revert Enterprise Replication and the **syscdr** database to an earlier version. These scripts write conversion and reversion messages for Enterprise Replication to the following locations:

- Output of the **concdr.sh** or **concdr.bat** script, which is standard output by default
- **concdr.out** file
- Output of the **revcdr.sh** or **revcdr.bat** script, which is standard output by default
- **revcdr.out** file
- **revtestcdr.out** file

You can find the **concdr.out**, **revcdr.out**, and **revtestcdr.out** files in **$INFORMIXDIR/etc** on UNIX or **%INFORMIXDIR%\etc** on Windows. For more information on converting and reverting Enterprise Replication, see the *IBM Informix Migration Guide*.

CDR reversion test completed successfully.

Cause: The **syscdr** database is revertible.

Action: None required.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

CDR reversion test failed; for details look in $INFORMIXDIR/etc/revtestcdr.out.

Cause: Enterprise Replication is not revertible.

Action: For more information, look at the messages in **revtestcdr.out**. Fix the reported problem before you attempt reversion.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

Enterprise Replication is not ready for conversion. The Control and TRG send queues should be empty for conversion/reversion to proceed.

Cause: There are elements in the control and Transaction Send Queue (also called TRG) send queues. The database server sends replicated data to the TRG queue before sending it to the target system.

Action: Wait for these queues to empty before you attempt either conversion or reversion. For more information, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

Prints this message to **concdr.out** during conversion or to **revcdr.out** during reversion.

Enterprise Replication is not ready for conversion. The syscdr database should NOT contain old-style group definitions for conversion to succeed.

Cause: The **syscdr** database *should not* contain old-style group definitions for conversion to succeed.

Action: Use the **cdr delete group** command to delete the old-style groups before attempting conversion. For more information, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

Prints this message to **concdr.out**.

Enterprise Replication should be in a stopped state for conversion/reversion to proceed.

Cause: Enterprise Replication should be in a stopped state for conversion or reversion to proceed.

Action: Stop Enterprise Replication. For more information, see the *IBM Informix Dynamic Server Enterprise Replication Guide*.

Prints this message to **concdr.out** during conversion or to **revcdr.out** during reversion.

Reversion of 'syscdr' failed; for details look in $INFORMIXDIR/etc/revcdr.out.

Cause: The reversion of the syscdr database failed.

Action: Find the cause of failure in the **revcdr.out** file, then fix the problem before you attempt reversion.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

Starting CDR reversion test...

Cause: This message displays at the beginning of Enterprise Replication reversion testing.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

Action: None required.

Starting ’syscdr’ conversion...

Cause: This message displays when you run the **concdr.sh** or **concdr.bat** script to convert the **syscdr** database to Version 10.0.

Action: None required.

Prints the output of the **concdr.sh** or **concdr.bat** script to standard output.

Starting ‘syscdr’ reversion...

Cause: This message displays when you run the **revcdr.sh** or **revcdr.bat** script to revert the **syscdr** database to an earlier version.

Action: None required.

Prints the output of the **revcdr.sh** or **revcdr.bat** script to standard output.

’syscdr’ conversion completed successfully.

Cause: This message displays after you complete converting Enterprise Replication and the **syscdr** database to Version 10.0.

Action: None required.

Prints the output of the **concdr.sh** or **concdr.bat** script to standard output.

’syscdr’ conversion failed. For details, look in $INFORMIXDIR/etc/concdr.out.

Cause: Conversion of the **syscdr** database failed.

Action: If conversion fails, resolve the problem reported in **concdr.out**. Restore the **syscdr** database from backup and reattempt conversion.

Prints the output of the **concdr.sh** or **concdr.bat** script to standard output.

Syscdr should NOT contain new replicate sets for reversion to succeed.

Cause: The new replicate sets in the **syscdr** database are not compatible with older versions.

Action: Use the **cdr delete replicateset** command to delete the replicate sets. Then rerun the **revcdr.sh** or **revcdr.bat** script to reattempt reversion.

Prints this message to **revtestcdr.out**.

Syscdr should not contain replicates defined with the --floatieee option for reversion to succeed.

Cause: Replicates have been defined with the **--floatiee** option. You cannot revert these replicates to the older version.

Action: Use the **cdr delete replicateset** command to delete replicates defined with the **--floatiee** option, then reattempt reversion.

Prints this message to **revtestcdr.out**.

Dynamic Log Messages

Dynamically added log file *logid* to DBspace *dbspace_number*.

Cause: The next active log file contains records of an open transaction. Whenever the database server adds a log dynamically, it logs this message. Example: `Dynamically added log file 38 to DBspace 5.`

Action: Complete the transaction as soon as possible.

Log file *logid* added to DBspace *dbspace_number*.

Cause: Whenever the administrator adds a log file manually, the database server logs this message. Example: `Log file 97 added to Dbspace 2.`

Action: None required.

Log file number *logid* has been dropped from DBspace *dbspace_number*.

Cause: When you drop a newly-added log file, the database server logs this message. Example: `Log file number 204 has been dropped from DBspace 17.`

Action: None required.

Log file *logid* has been pre-dropped.

Cause: When you drop a used log file, it is marked as deleted (status **D**) and cannot be used again. After you perform a level-0 backup, the database server drops this log file and can reuse the space. Example: `Log file 12 has been pre-dropped.`

Action: To delete the log file, perform a level-0 backup of all storage spaces.

Pre-dropped log file number *logid* has been deleted from DBspace *dbspace_number*.

Cause: After a backup, the database server deletes a pre-dropped log file and logs this message. Example: `Pre-dropped log file number 12 has been deleted from DBspace 3.`

Action: None required.

ALERT: Because the oldest logical log (*logid*) contains records from an open transaction (*transaction_address*), the server is attempting to dynamically add a log file. But there is no space available. Please add a DBspace or chunk. Then complete the transaction as soon as possible.

Cause: If the database server is unable to dynamically add a log file because the instance is out of space, it logs this message.

Action: Add a dbspace or chunk to an existing dbspace. Then complete the transaction as soon as possible.

ALERT: The oldest logical log (*logid*) contains records from an open transaction (*transaction_address*). Logical logging will remain blocked until a log file is added. Add the log file with the onparams -a command, using the -i (insert) option, as in:
onparams -a -d *dbspace* -s *size* -i
Then complete the transaction as soon as possible.

Cause: If the DYNAMIC_LOGS parameter is set to 1, the database server prompts the administrator to add log files manually when they are needed.

Action: Use the **onparams -a** command with the **-i** option to add the log file after the current log file. Then complete the transaction as soon as possible.

Log file *logid* has been pre-dropped. It will be deleted from the log list and its space can be reused once you take level-0 archives of all BLOBspaces, Smart BLOBspaces and non-temporary DBspaces.

Cause: When you drop a used log file, it is marked as deleted (status **D**) and cannot be used again, and **onparams** prints this message.

Action: To delete the log file, perform a level-0 backup of all storage spaces.

Sbspace Metadata Messages

Allocated *number* pages to Metadata from chunk *number*.

Cause: The database server freed the specified number of pages from the reserved area and moved them to the metadata area of chunk *number*.

Action: None required.

Allocated *number* pages to Userdata from chunk *number*.

Cause: The database server freed the specified number of pages from the reserved area and moved them to the user-data area of chunk *number*.

Action: None required.

Freeing reserved space from chunk *number* to Metadata.

Cause: The metadata area in chunk *number* is full. The database server is trying to free space from the reserved area to the metadata area.

Action: None required.

Freeing reserved space from chunk *number* to Userdata.

Cause: The user-data area in chunk *number* is full. The database server is trying to free space from the reserved area to the user-data area.

Action: None required.

Truncate Table Messages

The table cannot be truncated if it has an open cursor or dirty readers.

Cause: You must have exclusive access to the table.

Action: Wait for dirty readers to complete or close all the open cursors and reissue the TRUNCATE TABLE command.

The table cannot be truncated. It has at least one non-empty child table with referential constraints.

Cause: You cannot truncate a table if it has child tables with referential constraints and at least one row.

Action: Empty the child tables before you truncate this table.

Appendix F. Accessibility

The syntax diagrams in the HTML version of this manual are available in dotted decimal syntax format, which is an accessible format that is available only if you are using a screen reader.

Dotted Decimal Syntax Diagrams

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), the elements can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, `3` or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that your screen reader is set to read punctuation. All syntax elements that have the same dotted decimal number (for example, all syntax elements that have the number `3.1`) are mutually exclusive alternatives. If you hear the lines `3.1 USERID` and `3.1 SYSTEMID`, your syntax can include either `USERID` or `SYSTEMID`, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number `3` is followed by a series of syntax elements with dotted decimal number `3.1`, all the syntax elements numbered `3.1` are subordinate to the syntax element numbered `3`.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, the word or symbol is preceded by the backslash (\) character. The `*` symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element `*FILE` with dotted decimal number 3 is read as `3 \* FILE`. Format `3* FILE` indicates that syntax element `FILE` repeats. Format `3* \* FILE` indicates that syntax element `* FILE` repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol that provides information about the syntax elements. For example, the lines `5.1*`, `5.1 LASTRUN`, and `5.1 DELETE` mean that if you use more than one of the `LASTRUN` and `DELETE` syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the `%` symbol, this identifies a reference that is defined elsewhere. The string following the `%` symbol is the name of a syntax fragment rather than a literal. For example, the line `2.1 %OP1` means that you should refer to a separate syntax fragment `OP1`.

The following words and symbols are used next to the dotted decimal numbers:

? Specifies an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element (for example, 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

* Specifies a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data-area, you know that you can include more than one data area or you can include none. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

+ Specifies a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times. For example, if you hear the line 6.1+ data-area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. As for the * symbol, you can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

AIX; DB2; DB2 Universal Database; Distributed Relational Database Architecture; NUMA-Q; OS/2, OS/390, and OS/400; IBM Informix®; C-ISAM®; Foundation.2000™; IBM Informix ® 4GL; IBM Informix®DataBlade®Module; Client SDK™; Cloudscape™; Cloudsync™; IBM Informix®Connect; IBM Informix®Driver for JDBC; Dynamic Connect™; IBM Informix®Dynamic Scalable Architecture™(DSA); IBM Informix®Dynamic Server™; IBM Informix®Enterprise Gateway Manager (Enterprise Gateway Manager); IBM Informix®Extended Parallel Server™; i.Financial Services™; J/Foundation™; MaxConnect™; Object Translator™; Red Brick™; IBM Informix® SE; IBM Informix® SQL; InformiXML™; RedBack®; SystemBuilder™; U2™; UniData®; UniVerse®; wintegrate®are trademarks or registered trademarks of International Business Machines Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Windows, Windows NT, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names used in this publication may be trademarks or service marks of others.

Index

Special characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

P

Q

R

U

V

W

X

Printed in USA

G251-2268-01

Spine information:

IBM Informix **Version 10.0**

IBM Informix Dynamic Server Administrator's Reference