

DB/Cockpit

User's Manual

Version 9.2
September 1999
Part No. 000-6206

Published by Informix® Press

Informix Corporation
4100 Bohannon Drive
Menlo Park, CA 94025-1032

© 1999 Informix Corporation. All rights reserved. The following are trademarks of Informix Corporation or its affiliates, one or more of which may be registered in the United States or other jurisdictions:

Answers OnLine™; C-ISAM®; Client SDK™; DataBlade®; Data Director™; Decision Frontier™; Dynamic Scalable Architecture™; Dynamic Server™; Dynamic Server™, Developer Edition™; Dynamic Server™ with Advanced Decision Support Option™; Dynamic Server™ with Extended Parallel Option™; Dynamic Server™ with MetaCube®; Dynamic Server™ with Universal Data Option™; Dynamic Server™ with Web Integration Option™; Dynamic Server™, Workgroup Edition™; Dynamic Virtual Machine™; Enterprise Decision Server™; Formation™; Formation Architect™; Formation Flow Engine™; Gold Mine Data Access®; IIF.2000™; i.Reach™; i.Sell™; Illustra®; Informix®; Informix® 4GL; Informix® InquireSM; Informix® Internet Foundation.2000™; InformixLink®; Informix® Red Brick® Decision Server™; Informix Session Proxy™; Informix® Vista™; InfoShelf™; Interforum™; I-Spy™; Mediazation™; MetaCube®; NewEra™; ON-Bar™; OnLine Dynamic Server™; OnLine/Secure Dynamic Server™; OpenCase®; Orca™; PaVER™; Red Brick® and Design; Red Brick® Data Mine™; Red Brick® Mine Builder™; Red Brick® Decisionscape™; Red Brick® Ready™; Red Brick Systems®; Regency Support®; Rely on Red BrickSM; RISQL®; Solution DesignSM; STARindex™; STARjoin™; SuperView®; TARGETindex™; TARGETjoin™; The Data Warehouse Company®; The one with the smartest data wins.™; The world is being digitized. We're indexing it.SM; Universal Data Warehouse Blueprint™; Universal Database Components™; Universal Web Connect™; ViewPoint®; Visionary™; Web Integration Suite™. The Informix logo is registered with the United States Patent and Trademark Office. The DataBlade logo is registered with the United States Patent and Trademark Office.

Documentation Team: Kathy Eckardt, Laura Kremers, Richelle White

GOVERNMENT LICENSE RIGHTS

Software and documentation acquired by or for the US Government are provided with rights as follows:

- (1) if for civilian agency use, with rights as restricted by vendor's standard license, as prescribed in FAR 12.212;
- (2) if for Dept. of Defense use, with rights as restricted by vendor's standard license, unless superseded by a negotiated vendor license, as prescribed in DFARS 227.7202. Any whole or partial reproduction of software or documentation marked with this legend must reproduce this legend.

Table of Contents

Introduction

In This Introduction	3
About This Manual	3
Types of Users	3
Software Dependencies	4
Assumptions About Your Locale.	4
Demonstration Databases	4
New Features	5
Documentation Conventions	5
Typographical Conventions	6
Icon Conventions	7
Command-Line Conventions	7
Screen-Illustration Conventions	10
Additional Documentation	10
On-Line Manuals	11
Printed Manuals	11
Error Message Documentation	11
Documentation Notes, Release Notes, Machine Notes	12
Related Reading	12
Compliance with Industry Standards	12
Informix Welcomes Your Comments	13

Chapter 1	DB/Cockpit Overview	
	In This Chapter	1-3
	DB/Cockpit Architecture	1-4
	DB/Cockpit Components	1-4
	DB/Cockpit Tools	1-5
	Starting and Ending DB/Cockpit	1-7
	Before You Launch DB/Cockpit	1-7
	Launching onprobe	1-8
	Ending onprobe	1-9
	Launching oncockpit	1-10
	Ending oncockpit.	1-11
	The DB/Cockpit Main Window	1-12
 Chapter 2	 Using Alarms	
	In This Chapter	2-3
	Permanent and Session Alarm Files	2-4
	The Alarm Editor	2-6
	Opening the Alarm Editor	2-7
	Switching Between Alarm File Modes	2-8
	Editing Specific Alarms	2-8
	Defining a New Alarm	2-9
	Modifying an Existing Alarm	2-17
	Deleting an Existing Alarm	2-17
	DB/Cockpit Alarm Syntax	2-17
	DB/Cockpit Select List Syntax	2-18
	DB/Cockpit Criteria Syntax	2-21
	Field-IDs	2-22
	Managing Alarm Files	2-25
	Opening an Alarm File.	2-25
	Saving an Alarm File	2-25
	Replacing the Active Session Alarm File.	2-26
	Activating Alarm Files	2-26
	Preparing the Template Alarm Files for Activation	2-27
	Activating the Default Alarm Files.	2-29
	Activating Other Alarm Files	2-29
	The Alarm Display Window	2-30

Chapter 3

The DB/Cockpit Viewer

In This Chapter	3-3
Using the Drill-Down Method	3-4
Viewing Long Identifiers	3-4
Refreshing Information in Views	3-5
The DB/Cockpit Main Window	3-5
The Spaces View	3-8
The Space Information View	3-10
The Chunk Spaces View	3-12
The Chunk Space Information View.	3-13
The Chunk Disk Fragmentation View	3-15
The Databases View	3-18
The Database Information View	3-20
The Tables View	3-22
The Table Profile View	3-24
The Table Information View	3-26
The Table Disk Fragmentation View.	3-28
The Sessions View	3-30
The Session Profile View.	3-32
The SQL Statement View	3-35
The Logical Logs View	3-37
The Physical Log View	3-39
The Data Replication View	3-41
The Virtual Processors View	3-43
The Active VPs View	3-45
The VP Information View	3-46
The I/O VP Queues View	3-49
The Shared Memory View	3-51
The SHM Buffers View	3-54
The SHM Segments View	3-56
The ISAM View.	3-58
The Overflow View	3-59
The Waits View	3-60
The Disk View	3-62

Chapter 4	The Severity Analyst	
	In This Chapter	4-3
	How Does the Severity Mechanism Operate?	4-3
	Managing Severities	4-4
	Defining Severity Expressions	4-4
	Assigning Colors to Severity Levels	4-6
	Loading a Severity File.	4-6

Index

Introduction

In This Introduction	3
About This Manual.	3
Types of Users	3
Software Dependencies	4
Assumptions About Your Locale.	4
Demonstration Databases	4
New Features.	5
Documentation Conventions	5
Typographical Conventions	6
Icon Conventions	7
Command-Line Conventions	7
How to Read a Command-Line Diagram	9
Screen-Illustration Conventions	10
Additional Documentation	10
On-Line Manuals	11
Printed Manuals	11
Error Message Documentation	11
Documentation Notes, Release Notes, Machine Notes	12
Related Reading	12
Compliance with Industry Standards	12
Informix Welcomes Your Comments.	13

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 summarizes the features that make up the DB/Cockpit utility, a graphical tool designed especially for the following Informix Dynamic Server 2000 administrative tasks: managing Dynamic Server, maintaining its integrity, and ensuring a smooth-running database for other users.

Types of Users

This manual is written for database server administrators.

This manual assumes 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
- Working knowledge of Structured Query Language (SQL)

If you have limited experience with relational databases, SQL, or your operating system, refer to the [Getting Started](#) manual for your database server for a list of supplementary titles.

Software Dependencies

This manual assumes that you are using Informix Dynamic Server 2000 on UNIX.

Assumptions About Your Locale

Informix products can support many languages, cultures, and code sets. All culture-specific information 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 dates, times, 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 [Informix Guide to GLS Functionality](#).

Demonstration Databases

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 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 *DB-Access User Manual*. For descriptions of the databases and their contents, see the *Informix Guide to SQL: Reference*.

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

New Features

For a comprehensive list of new features for your database server, see the release notes.

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
- Icon conventions
- Command-line conventions
- Screen illustration conventions

Typographical Conventions

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




Convention	Meaning
KEYWORD	All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font.
<i>italics</i> italics <i>italics</i>	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 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.
<code>monospace</code> <code>monospace</code>	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 the end of one or more product- or platform-specific paragraphs.
→	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.

Icon Conventions

Throughout the documentation, you will find text that is identified by several different types of icons. This section describes these icons.

Icon	Label	Description
	<i>Warning:</i>	Identifies paragraphs that contain vital instructions, cautions, or critical information
	<i>Important:</i>	Identifies paragraphs that contain significant information about the feature or operation that is being described
	<i>Tip:</i>	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described

Command-Line Conventions



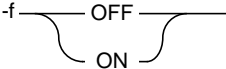
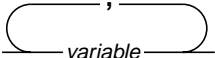
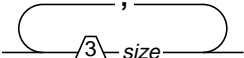
This section defines and illustrates the format of commands that are available in Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

Each diagram displays the sequences of required and optional elements that are valid in a command. A diagram begins at the upper-left corner with a command. It ends at the upper-right corner with a vertical line. Between these points, you can trace any path that does not stop or back up. Each path describes a valid form of the command. You must supply a value for words that are in italics.

You might encounter one or more of the following elements on a command-line path.

Element	Description
command	This required element is usually the product name or other short word that invokes the product or calls the compiler or preprocessor script for a compiled Informix product. It might appear alone or precede one or more options. You must spell a command exactly as shown and use lowercase letters.
<i>variable</i>	A word in italics represents a value that you must supply, such as a database, file, or program name. A table following the diagram explains the value.
-flag	A flag is usually an abbreviation for a function, menu, or option name or for a compiler or preprocessor argument. You must enter a flag exactly as shown, including the preceding hyphen.
.ext	A filename extension, such as .sql or .cob , might follow a variable that represents a filename. Type this extension exactly as shown, immediately after the name of the file. The extension might be optional in certain products.
(, , ; + * - /)	Punctuation and mathematical notations are literal symbols that you must enter exactly as shown.
' '	Single quotes are literal symbols that you must enter as shown.
<div>Privileges p. 5-17</div> <div>Privileges</div>	A reference in a box represents a subdiagram. Imagine that the subdiagram is spliced into the main diagram at this point. When a page number is not specified, the subdiagram appears on the same page.
<div>Table Name see Database Object Name in SQLS</div>	A reference to SQLS in this manual refers to the <i>Informix Guide to SQL: Syntax</i> . Imagine that the subdiagram is spliced into the main diagram at this point.
<div>— ALL —</div>	A shaded option is the default action.

(1 of 2)

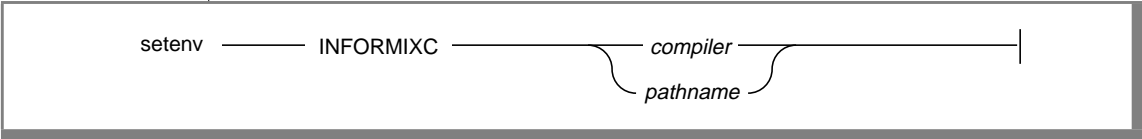
Element	Description
	Syntax within a pair of arrows indicates a subdiagram.
	The vertical line terminates the command.
	A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.)
	A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items.
	A gate ($\sqrt{3}$) on a path indicates that you can only use that path the indicated number of times, even if it is part of a larger loop. Here you can specify <i>size</i> no more than three times within this statement segment.

(2 of 2)

How to Read a Command-Line Diagram

Figure 1 shows a command-line diagram that uses some of the elements that are listed in the previous table.

Figure 1
Example of a Command-Line Diagram



To construct a command correctly, start at the top left with the command. Then follow the diagram to the right, including the elements that you want. The elements in the diagram are case sensitive.

Figure 1 on page 9 diagrams the following steps:

1. Type the word `setenv`.
2. Type the word `INFORMIXC`.
3. Supply either a compiler name or pathname.
After you choose *compiler* or *pathname*, you come to the terminator.
Your command is complete.
4. Press RETURN to execute the command.

Screen-Illustration Conventions

The illustrations in this manual represent a generic rendition of various windowing environments. The details of dialog boxes, controls, and windows were deleted or redesigned to provide this generic look. Therefore, the illustrations in this manual depict the windowing environment a little differently than the way it appears on your screen.

Additional Documentation

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

- On-line manuals
- Printed manuals
- On-line help
- Error message files
- Documentation notes, release notes, and machine notes
- Related reading

On-Line Manuals

An Answers OnLine CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print on-line manuals, see the installation insert that accompanies Answers OnLine.

Informix on-line manuals are also available on the following Web site:

`www.informix.com/answers`

Printed Manuals

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com. Please provide the following information when you place your order:

- The documentation that you need
- The quantity that you need
- Your name, address, and telephone number

Error Message Documentation

Informix software products provide ASCII files that contain all of the Informix error messages and their corrective actions.

To read error messages and corrective actions, use one of the following utilities.

Utility	Description
finderr	Displays error messages on line
rofferr	Formats error messages for printing

Documentation Notes, Release Notes, Machine Notes

In addition to printed documentation, the following sections describe the on-line files that supplement the information in this manual. Please examine these files before you begin using your database server. They contain vital information about application and performance issues.

The following on-line files appear in the `$INFORMIXDIR/release/en_us/0333` directory.

On-Line File	Purpose
COCKPITDOC_9.2	The documentation-notes file describes features that are not covered in this manual or that have been modified since publication.
SERVERS_9.2	The release-notes file describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.
IDS_9.2	The machine-notes file describes any special actions that are required to configure and use Informix products on your computer. Machine notes are named for the product described.

Related Reading

For a list of publications that provide an introduction to database servers and operating-system platforms, refer to your *Getting Started* manual.

Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. 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 Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

Informix Welcomes Your Comments

Let us know what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about any corrections or clarifications that you would find useful. Include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

Send electronic mail to us at the following address:

`doc@informix.com`

The **doc** alias is reserved exclusively for reporting errors and omissions in our documentation.

We appreciate your suggestions.

DB/Cockpit Overview

In This Chapter	1-3
DB/Cockpit Architecture.	1-4
DB/Cockpit Components	1-4
DB/Cockpit Tools	1-5
Starting and Ending DB/Cockpit	1-7
Before You Launch DB/Cockpit	1-7
Launching onprobe	1-8
Ending onprobe	1-9
Launching oncockpit	1-10
Ending oncockpit	1-11
The DB/Cockpit Main Window	1-12

In This Chapter

This chapter introduces DB/Cockpit, a graphical tool designed especially for the following Informix Dynamic Server 2000 administrative tasks: monitoring Dynamic Server, maintaining its integrity, and ensuring a smooth running database for other users.

DB/Cockpit allows you to monitor many Dynamic Server system parameters that other utilities report. It has the following advantages:

- It is a graphic environment and not just a graphics displayer.
- It shows actual, numeric values.
- Data presentation is intuitive and straightforward.

DB/Cockpit provides you with the right decision-making tools for gathering information about Dynamic Server. You can use this information to solve problems and improve Dynamic Server performance.

DB/Cockpit Architecture

This section describes components and tools, which are the two aspects of DB/Cockpit architecture.

DB/Cockpit Components

DB/Cockpit implements a client/server architecture. It combines two major components, the **onprobe** server and the **oncockpit** client. The **onprobe** program is the data collector; the **oncockpit** program is responsible for user interaction.

The **onprobe** program receives *requests* for information about Dynamic Server status and activity. The **onprobe** program handles two types of requests:

- Regular requests forwarded from **oncockpit** clients
- Conditional requests called *alarms*, which both **onprobe** and **oncockpit** can issue

The **onprobe** program samples the required data according to the requests received and at predefined rates. It always replies to regular requests by returning the corresponding data. If **onprobe** receives an alarm, it only returns data when the condition of the alarm proves true. The **onprobe** program evaluates the severity of all the data it returns according to predefined severity expressions.

The **onprobe** program extracts the requested data from two sources: the System Monitoring Interface (SMI) database and directly from Dynamic Server shared memory (SHM).

The **oncockpit** program is a two-way highway. You use it to send requests for information about current database activity and configuration. You also use it to receive the data returned by your requests and display any alarms directed to it by **onprobe**.

A dual process architecture has two major advantages:

- You can monitor any database server directly from your workstation without overtaxing database server resources and at the same time, benefit from a graphics environment. (Many database servers do not have an X display.) Only **onprobe** resides on the same database server as the monitored Dynamic Server system. The **oncockpit** program can reside on any host computer in the network as well as on the database server itself.
- The **onprobe** program can function independently, without **oncockpit**, and serve sites that do not have a resident Dynamic Server administrator. The **onprobe** program watches over the system, issuing alarms when needed and recording user-defined events.

DB/Cockpit Tools

DB/Cockpit provides you with the following three tools:

- Alarms
You can customize DB/Cockpit to warn against exceptional system behavior and other anomalies when they occur and directly on the screen. You select the parameters that you want to monitor, and DB/Cockpit triggers an alarm when one of them reaches a certain threshold. Performance related events and peak-hour anomalies can all be detected and analyzed.
- Viewer
The Viewer is an information collector and a data evaluator. It presents real-time information about system configuration, system operation, disk and data layout, current activities, and user and resource management. Each subject is displayed in a separate *View*. Each View is a window and, like all true windows, you can open as many views as you need on your screen.



Important: One major difference exists between alarms and views. Information retrieved for alarms is user dependent. Views are predefined to show specific information.

■ **Severity Analyst**

You can receive out-of-range alerts about the values you are browsing should one of them deviate from the range you defined. If you set normal ranges for system parameters that interest you, DB/Cockpit passes their data through an evaluator that uses a color-coded mechanism to emphasize anything out-of-range.

DB/Cockpit supports the following five severity levels:

- ❑ Warning
- ❑ Error
- ❑ Fatal
- ❑ Undefined
- ❑ Normal (which plays a chief role)

Color-coded data is not only easily interpreted, it also assists in on-the-spot analysis and system tuning.

Severities have one other on-line advantage. You can color code the data that the Viewer presents (or simply put, you can see most deviations on time and act on time). You can also base alarm definition on severity levels, making sure that important values initiate alarms when they cross the severity threshold defined for them.

Starting and Ending DB/Cockpit

This section details how to launch and end DB/Cockpit.

Before You Launch DB/Cockpit

The **onprobe** and **oncockpit** programs communicate with each other through a TCP *service*. Therefore, before you can launch either **onprobe** or **oncockpit**, you should define the service.

To define a suitable service

- If you use NIS, add a new TCP service to the NIS server **services** map.
- If you do not use NIS, add an identical TCP service to your **/etc/services** files on both the **onprobe** and **oncockpit** computers.

Make sure to assign a unique service number, as the following example shows:

```
cockpit      1555/tcp # DB/Cockpit
```

During launching, **onprobe** and **oncockpit** search for the service name according to the following precedence:

1. The **-service** command-line option.
2. The **COCKPITSERVICE** environment variable.

The following example illustrates settings for the **COCKPITSERVICE** environment variable in the C shell and the Bourne (or Korn) shell:

C shell: `setenv COCKPITSERVICE cockpit2`

Bourne shell: `COCKPITSERVICE=cockpit2
export COCKPITSERVICE`

For more information about setting environment variables, refer to your UNIX documentation.

3. The default, which is set to **cockpit**, is assumed if you do not specify the service name either in the command line or as an environment variable.



4. Environment variables that start with `_COCKPIT_` are for internal Informix use. They are used for problem-solving purposes, thus use of such variables might cause a change in the behavior of DB/Cockpit.

Important: If you have multiple Dynamic Server instances on the same computer, define one service per Dynamic Server instance (because each **onprobe** instance monitors a different Dynamic Server instance).

Launching onprobe

To launch **onprobe**, you enter various command-line options at the operating-system prompt.

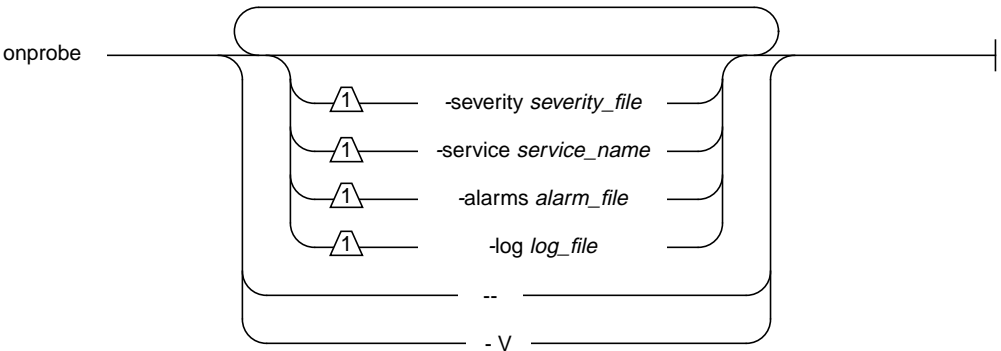
To launch onprobe

1. Log in as user **informix** or **root**.
2. Make sure your **INFORMIXDIR** and **INFORMIXSERVER** environment variables are set exactly like those of the monitored Dynamic Server system.



Important: For performance reasons, Informix recommends that you use **onipcshm** as the connection type in the `$INFORMIXDIR/etc/sqlhosts` file for the **INFORMIXSERVER** intended for use by **onprobe**.

3. Optionally, set the **COCKPITSERVICE** environment variable, as the previous section explains.
4. Run **onprobe** using the following syntax.



-severity	Load the severity file <i>severity_file</i> . \$INFORMIXDIR/etc is added to any filename that does not begin with a dot (.) or a slash (/). The default file is \$INFORMIXDIR/etc/severity .
-service	Accept oncockpit clients using the service <i>service_name</i> . The default service is named cockpit .
-alarms	Load the permanent alarm file <i>alarm_file</i> . \$INFORMIXDIR/etc is added to any filename that does not begin with a dot (.) or a slash (/). The default file is \$INFORMIXDIR/etc/permalrm .
-log	Append log messages to the file <i>log_file</i> . If you use a hyphen (-) to designate <i>log_file</i> , the log messages will be printed to the standard output.
--	Print this usage message.
-V	Print the current product version.



***Tip:** You can add the sequence that launches **onprobe** to the suitable **rc** file, after the **oninit** command that starts the monitored Dynamic Server system.*

Ending onprobe

The **onprobe** program is a UNIX daemon process. If you want to end **onprobe**, use the **Kill** command.

However, when you change the Dynamic Server mode to quiescent or off-line, you do not have to end **onprobe** because it detects the Dynamic Server mode automatically. Although **onprobe** monitors a Dynamic Server instance, it continues functioning by itself even when Dynamic Server is off-line, waiting until it is restarted.

While Dynamic Server is in off-line mode, **onprobe** cannot extract data. When requested for data, it sends an error message instead. However, while Dynamic Server is in quiescent mode, **onprobe** can extract data from shared memory but not from SMI.



Launching oncockpit

To launch **oncockpit**, you enter various command-line options at the operating-system prompt.

Important: As a Motif application, **oncockpit** uses a runtime file **XKeysymDB**. This file contains keyboard keysym-keycode mappings and is a part of the X11/Motif installation. The placement and contents of this file vary depending on platform vendor and particular X11/Motif installation. The most commonly used file locations are:

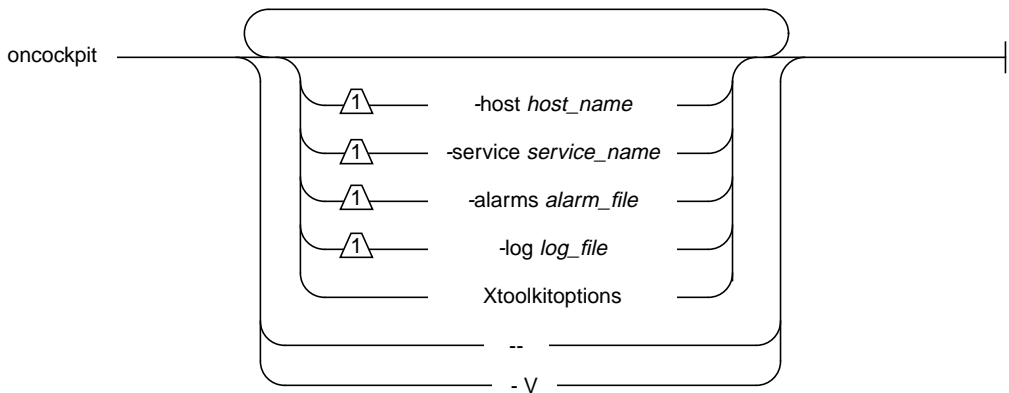
/usr/lib/X11/XKeysymDB

/usr/X11R/lib/X11/XKeysymDB

/usr/openwin/lib/X11/KeysymDB (SunOS and Solaris only)

To launch oncockpit

1. Optionally, set the **COCKPITSERVICE** environment variable to the suitable service name.
2. Run **oncockpit** using the following syntax.



- host** Connect to the **onprobe** server running on the host *host_name*. The default is the local host.
- service** Connect to the **onprobe** server using the service *service_name*. The default service is named **cockpit**.
- alarms** Load the session alarm file *alarm_file*. **\$INFORMIXDIR/etc** is added to any filename that does not begin with a dot (.) or a slash (/). The default file is **\$INFORMIXDIR/etc/sessalarm**.
- log** Append log messages to the file *log_file*. If you use a hyphen (-) to designate *log_file*, the log messages will be printed to the standard output.
- Xtoolkitoptions** Any standard Xt options.
- Print this usage message.
- V** Print the current product version.



Important: Whether you set the service name as an environment variable or specify it in the command line, make sure you use the service name that is assigned to the **onprobe** server to which you want to connect.

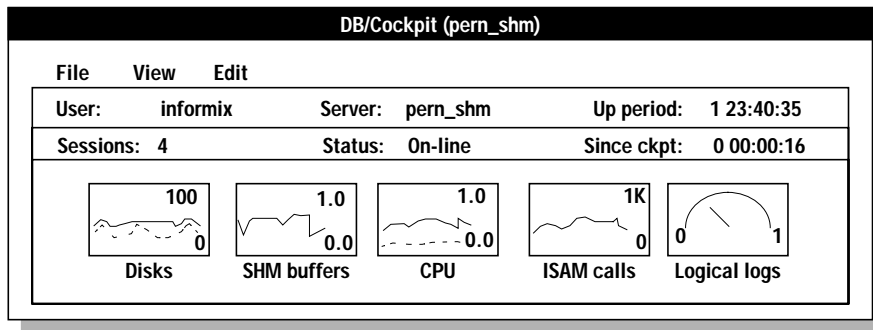
Ending oncockpit

To end **oncockpit**, choose **File→Exit** from the Main window.

The DB/Cockpit Main Window

When you launch **oncockpit**, the DB/Cockpit main window is displayed, as [Figure 1-1](#) shows.

Figure 1-1
The DB/Cockpit Main Window



The lower part of the DB/Cockpit main window presents graphical displays of the resources that the Dynamic Server system uses:

- The solid line on the **Disks** graph shows total disk operations (reads and writes) that the Dynamic Server system performs per second. The broken line shows disk writes. The distance between the two lines represents disk reads.
- The **SHM buffers** graph shows the percentage of dirty shared memory buffers.
- The solid line on the **CPU** graph shows total CPU time consumed by all user threads in the Dynamic Server system. The broken line shows system CPU consumed by all user threads in the Dynamic Server system. The distance between the two lines represents user CPU.
- The **ISAM calls** graph shows the total number of ISAM calls. This graph is a good indicator of the requirement for Dynamic Server services.
- The **Logical logs** graph gauges the percentage of logical logs currently used.

Using Alarms

In This Chapter	2-3
Permanent and Session Alarm Files	2-4
The Alarm Editor	2-6
Opening the Alarm Editor	2-7
Switching Between Alarm File Modes	2-8
Editing Specific Alarms	2-8
Defining a New Alarm	2-9
Message	2-9
Type and Field-IDs	2-10
Active.	2-11
SQL Text/Criteria	2-12
Sampling Frequency.	2-15
Reschedule	2-15
Deliver to	2-16
Modifying an Existing Alarm	2-17
Deleting an Existing Alarm.	2-17
DB/Cockpit Alarm Syntax	2-17
DB/Cockpit Select List Syntax	2-18
DB/Cockpit Criteria Syntax	2-21
Field-IDs	2-22
Managing Alarm Files.	2-25
Opening an Alarm File	2-25
Saving an Alarm File	2-25
Replacing the Active Session Alarm File	2-26
Activating Alarm Files	2-26
Preparing the Template Alarm Files for Activation	2-27
Activating the Default Alarm Files	2-29
Activating Other Alarm Files	2-29
The Alarm Display Window	2-30



In This Chapter

An *alarm* is a request that enables you to check for a specific condition at a given interval and warn when that condition is met. You can use alarms for almost anything, for example, to warn against dbspaces or logs that fill up as well as deadlocks when they occur.

***Tip:** The request is the basic communication tool between **onprobe** and its **oncockpit** clients. The **onprobe** program executes requests and replies with data collected from the monitored Dynamic Server system. The alarm is a special request because it has an auto-refresh mechanism that you can customize by setting how often to sample data and under what conditions. Although the viewer is also based on the auto-refresh mechanism, the end user cannot alter its request parameters.*

An *alarm file* is a set of alarms that can function at the same time. An alarm file can contain no alarms at all, one alarm, or as many alarms as you want. After you create the alarm file, you can change it by adding new alarms to it, deleting alarms from it, and modifying a certain alarm without affecting other alarms in the file.

You can create numerous alarm files. An alarm file can be dormant (file on the disk) or active (loaded in **onprobe**). You can switch between active alarm files as the need arises.

Permanent and Session Alarm Files

DB/Cockpit supports *permanent* alarm files and *session* alarm files.

- The permanent alarm file is an integral part of **onprobe** and is therefore loaded at **onprobe** start-up. By default, **onprobe** looks for a permanent alarm file named **permalrm** under **SINFORMIXDIR/etc**.

You can only have one active permanent alarm file at a given time, but you can have more than one dormant file. You are free to change the number of alarms in a file as well as their syntax and parameters. To change the permanent alarm file, you need to work from the **oncockpit** Alarm Editor. If you want to activate another alarm file or reactivate the active file after modifying it, you must restart **onprobe** for changes to take effect.

You can direct replies resulting from permanent alarms to a UNIX pipe, append them to an alarm log file, or send them to a desired address by way of electronic mail.

- The session alarm file originates in **oncockpit**. By default, **oncockpit** looks for a session alarm file named **sessalrm** under **SINFORMIXDIR/etc**.

You can customize a session alarm file and immediately activate it (without restarting either **onprobe** or **oncockpit**) by choosing a simple menu item. In addition, you can also create as many dormant session alarm files as your system dictates and activate the desired file as the need arises.

As with permanent alarms, replies from session alarms can be sent to a UNIX pipe, appended to an alarm log file, or sent by way of electronic mail to a certain address. In addition, because of their graphic origin, replies from session alarms can also be projected on the screen in real time.

How do different characteristics affect alarm functionality? Permanent alarms are suited for global Dynamic Server system subjects and tend to remain fixed. Permanent alarms serve as the system watchdog, on the lookout for major problems. For example, permanent alarms are ideal for monitoring logical logs: their absence points to a smooth-running system.

Session alarms, on the other hand, are easy to customize and replace. They can be directed to handle specific issues, answer everyday dynamic-type needs, and serve as local troubleshooters. For example, you can create a special alarm file to check caching.

Permanent and session alarm files can be active simultaneously: one and only one permanent alarm file from **onprobe**, and one session alarm file from each **oncockpit** client. The permanent alarm file is not dependent on a graphical interface and because it is executed directly from **onprobe**, you can use it at sites that do not have a full-time Dynamic Server administrator. Activating a session alarm file, on the other hand, requires a running **oncockpit** client.



***Important:** DB/Cockpit comes ready-made with templates of a permanent alarm file and a session alarm file. For more information on how to customize and activate these files, refer to “[Activating Alarm Files](#)” on page 2-26.*

The following table summarizes the two alarm file types.

Permanent Alarm File	Session Alarm File
Requires onprobe to function.	Requires oncockpit as well as onprobe to function.
By default, onprobe loads the file named permalrm from \$INFORMIXDIR/etc.	By default, oncockpit loads the file named sessalrm from \$INFORMIXDIR/etc.
The number of alarms in the file is flexible, depending on your needs.	The number of alarms in the file is flexible, depending on your needs.
Replies to alarms can be piped, logged in a file, or mailed.	Replies to alarms can be piped, logged in a file, mailed, or projected in real time on a graphic screen.
Can be edited from oncockpit .	Can be edited from oncockpit .
To load a new alarm file, you have to restart onprobe .	Loading a new alarm file is immediate and can be executed during runtime from oncockpit .
Serves as a watchdog for the Dynamic Server system.	Serves as on-the-spot troubleshooters.



The Alarm Editor

The Alarm Editor is an **oncockpit** tool that allows you to manage and modify session alarm files as well as permanent alarm files.

Important: *Do not mix the current alarm file displayed in the Alarm Editor with the active alarm files (permanent and session) loaded in **onprobe**.*

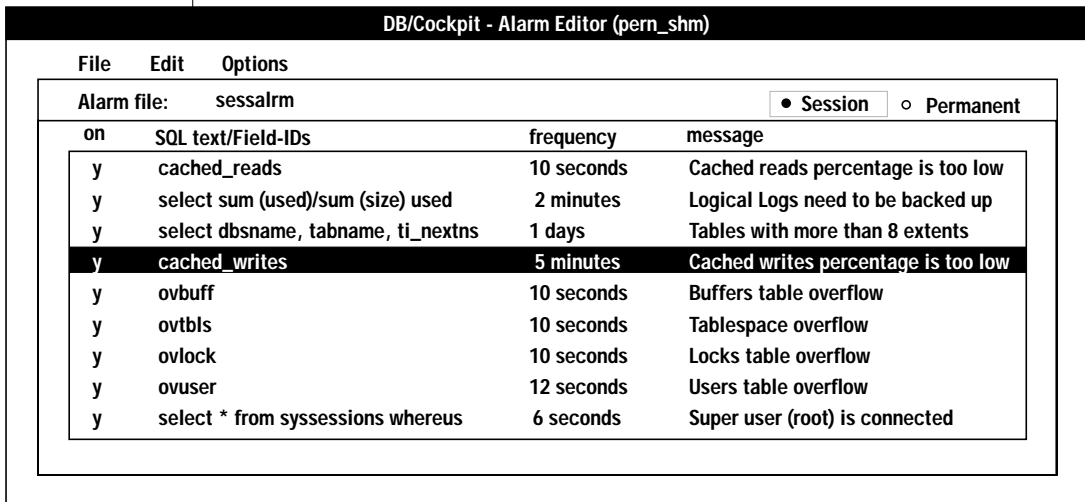
The Alarm Editor allows you to:

- open and display an existing alarm file.
- enter the Alarm Request dialog box where you can create a new alarm or modify an existing alarm within the current alarm file.
- delete an existing alarm from the file.
- save a new or modified alarm file.
- replace the active session alarm file with the alarm file currently displayed.

Opening the Alarm Editor

To open the Alarm Editor, choose **Edit→Alarms** from the main window of **oncockpit**. When you open the Alarm Editor for the first time, a list of the alarms defined in the active session alarm file is displayed, as [Figure 2-1](#) shows.

Figure 2-1
The Alarm Editor



The menu bar offers commands for managing specific alarms and the alarm file as a whole.

The name of the current alarm file is displayed above the alarm list, on the left side. The working mode for the current alarm file, session or permanent, is displayed on the right side.

The list includes an on/off check, alarm name, sampling frequency, and the message issued if the alarm is triggered.

Switching Between Alarm File Modes

The Alarm Editor supports two working modes, one for permanent alarm files and the other for session alarm files. Modes are displayed as radio buttons (**Session** and **Permanent**) above the message list on the right side.

Most options are supported in both modes. There are, however, two practical differences between the modes. In permanent mode, the **Options→Replace** menu item is grayed (because you cannot activate a permanent alarm file from **oncockpit**) and output to a graphic popup window is disabled.

When you enter the Alarm Editor for the first time, Session mode is turned on. On subsequent entries to the Alarm Editor, the last mode activated is restored. To change modes, click the desired option button. Remember to check whether the desired mode is turned on before you save an alarm file.

Editing Specific Alarms

An alarm includes the following components that you can edit:

- Message issued with alarm data, if the alarm is triggered
- Alarm type: SMI or Cockpit
- Active check box, used to turn a certain alarm on or off, without affecting the rest of the alarm file
- Conditions by which to filter Dynamic Server data. An alarm is triggered only when the conditions are met
- Sampling frequency that specifies how often to evaluate the alarm
- Rescheduling time, specifying when **onprobe** should resume alarm evaluation after the alarm was last triggered
- Output destination for alarm data and message if the alarm is triggered

The **Edit** menu offers the following items for editing the current file:

- Defining a new alarm
- Modifying the components of a specific alarm
- Deleting an alarm

Defining a New Alarm

To define a new alarm, choose **Edit**→**New** in the Alarm Editor. The Alarm Request dialog box opens, as [Figure 2-2](#) shows.

Figure 2-2
Alarm Request Dialog Box with Initial Settings

DB/Cockpit - Alarm Request (pern_shm)

Message: _____

Type and Field-IDs: _____

<input type="checkbox"/> SMI	<input checked="" type="checkbox"/> Active
<input checked="" type="checkbox"/> Cockpit	<input type="text"/> ...

Criteria: _____

▲

▼

◀

▶

Sampling frequency: _____

Every:

Reschedule: _____

☒ After:

☐ Never

Deliver to: _____

☒ Popup alarm

☐ Mail:

☐ Pipe:

☐ Log File:

Specify the following items in the Alarm Request dialog box.

Message

You can add any free text in this area. Text that you write is attached to the output when the alarm is triggered to help you understand the event that caused the alarm.

The following example shows text attached to an alarm that checks the usage of logical logs:

Time to back up logical logs

Type and Field-IDs

You can define an alarm in either of the following ways:

- An SMI alarm extracts data from the **sysmaster** database by using SQL SELECT statements.

To create an SMI alarm, click **SMI**. (Note that the title of the lower-left area of the dialog box changes from Criteria to SQL Text.)

If the alarm is triggered, SELECT statement output is returned.

- A Cockpit alarm uses data-retrieval procedures especially defined for DB/Cockpit.

To create a Cockpit alarm, click **Cockpit**.

The Cockpit alarm includes a *select list* that defines the data to be returned and *criteria* that define the condition that triggers the alarm.

The select list contains one or more *field-ID* based expressions whose values you want retrieved for examination if the alarm is triggered.

Specify the desired select list in the right text box. The select list resembles the SELECT list of a SELECT statement. For more information about select list syntax, refer to [“DB/Cockpit Select List Syntax” on page 2-18](#).

To specify a field-ID, either type it in or select it from the list box. To access the list box, click the **ellipsis** button on the right side of the text box.

To select a field-ID from the list, use one of the following methods:

- ❑ Double-click the desired field-ID in the list to insert it at the position of the cursor.
- ❑ Click the field-ID and then click **Apply**. The field-ID is inserted at the position of the cursor.
- ❑ Click the field-ID with the center mouse button and drag it to the desired place.

The following example represents an alarm that reports logical-log usage as a percentage:

```
loglogs_used * 100
```

The next example examines cache usage:

```
cached_reads, cached_writes
```

The last example reports the total size of Dynamic Server resident, virtual and message shared-memory segments, in bytes:

```
totshmem * 1024 shared_memory_size_in_bytes
```

Active

By default, every new alarm that you define is active, that is, **onprobe** executes it.

To deactivate an alarm so that it will not be executed with other alarms in the alarm file, click **Active** in the **Type and Field-IDs** group. This feature might come in handy when you want to tune a specific alarm on-line without being interrupted by the triggering of other alarms. You can also use this feature when you want to temporarily deactivate an alarm based on a heavy SQL statement.



Important: Note the difference between an active alarm file and an active alarm. An active alarm file is the file loaded in **onprobe**; an active alarm is an enabled alarm. Both an active alarm file and a dormant alarm file can include active and nonactive alarms.

SQL Text/Criteria

In this area, you define the condition on which the alarm is based:

- If you click **SMI**, specify the SQL text here, as [Figure 2-3](#) shows. You can only use SELECT statements and query only the **sysmaster** database. (The current database for SELECT statements is **sysmaster**.)

Type the SELECT statement in the **SQL text** text box. For more information about SELECT statements, refer to the [Informix Guide to SQL: Syntax](#).

The following statement triggers an alarm when **onprobe** detects one or more sessions that super user (root) runs. Session IDs are returned with other alarm data.

```
select sid from syssessions where username='root'
```

The statement in the second example triggers an alarm when more than one user is connected (the **onprobe** session is taken into account). The number of users is returned with other alarm data.

```
select count (*) from syssessions having count (*) > 2
```

The alarm in the last example notifies you about dbspaces that are more than 90 percent full:

```
select sysdbspaces.dbsnum, sysdbspaces.name,  
       (sum(chksize) - sum(nfree)) / sum(chksize)  
used,  
       sum(chksize) total_pages, sum(nfree) free_pages  
from syschunks, sysdbspaces  
where syschunks.dbsnum = sysdbspaces.dbsnum  
      and sysdbspaces.is_blobspace = 0  
      and syschunks.dbsnum in (select dbsnum  
                               from sysdbspaces)  
group by sysdbspaces.dbsnum, sysdbspaces.name  
having sum(nfree) / sum(chksize) < 0.1  
order by sysdbspaces.dbsnum
```

Figure 2-3
The Alarm Request Dialog Box That Shows an SMI Alarm

DB/Cockpit - Alarm Request (pern_shm)

Message:

Type and Field-IDs: ☒ SMI ☒ Active
☐ Cockpit ...

SQL text:

Sampling frequency: Every:

Reschedule: ☒ After:
☐ Never

Deliver to: ☒ Popup alarm
☒ Mail:
☒ Pipe:
☒ Log File:

- If you click **Cockpit**, define the criteria in the **Criteria** text box, as [Figure 2-4 on page 2-15](#) shows.

The criteria are field-ID based conditions evaluated by **onprobe**. When they prove true, the alarm is triggered.

The criteria resemble the WHERE clause of a SQL SELECT statement. For more information about syntax, refer to [“DB/Cockpit Criteria Syntax” on page 2-21](#).

Usually, any field-ID that appears in the select list in the **Type and Field-IDs** group also appears in the criteria.

To specify a field-ID, either type it in or select it from the list box. To access the list box, click the **ellipsis** button on the right side of the text box.

Examples:

```
loglogs_used > 0.5
```

The criterion defines a condition for logical-log overflow. To receive data about logical-log usage in percent, define `loglogs_used * 100` in the **Type and Field-IDs** group.

```
cached_reads < 0.95 OR cached_writes < 0.85
```

The criteria here examine cache usage, triggering an alarm when the percentage of cached reads falls below 0.95 or the percentage of cached writes falls below 0.85. To receive values for cached reads and writes, their field-IDs should appear in the **Type and Field-IDs** group.

```
slope(fgwrites) > 0
```

In this example, an alarm is triggered when a foreground write occurs. `Slope()` is a special DB/Cockpit function that measures changes in the sampled value over a period of time. To receive values for foreground writes, `fgwrites` should appear in the **Type and Field-IDs** group.

```
slope(deadlks)
```

The condition in this example triggers an alarm when deadlocks occur. To receive the actual number of deadlocks, define `deadlks` in the **Type and Field-IDs** group.

```
online_mode != "On-line"
```

The condition in this example instructs **onprobe** to warn when the database server is not in on-line mode. Here too, `online_mode` should appear in the **Type and Field-IDs** group to receive the current database server mode.

```
severity (loglogs_used) = "FATAL"
```

The condition in this example is based on the severity mechanism. The alarm will be triggered when the severity defined for logical-log usage reaches fatal level. For more information about severity expressions, refer to [Chapter 4, "The Severity Analyst."](#)

Tip: To define an alarm whose sole purpose is data retrieval, do not specify criteria. An alarm with no criteria is triggered as if its conditions proved true.



Figure 2-4
The Alarm Request Dialog Box That Shows a Cockpit Alarm

DB/Cockpit - Alarm Request (pern_shm)

Message:

Type and Field-IDs: ☐ SMI ☒ Active
☒ Cockpit ...

Criteria:

Sampling frequency: Every:

Reschedule: ☒ After:
☐ Never

Deliver to: ☒ Popup alarm
☒ Mail:
☒ Pipe:
☒ Log File:

Sampling Frequency

In the **Sampling frequency** group, set the **onprobe** sampling rate for the alarm (in seconds, minutes, hours, or days). Type the desired sampling frequency in the text box and then select a desired time unit from the list box.

Reschedule

Reschedule sets behavior strategy after an alarm is triggered. Once triggered, an alarm is automatically deactivated by **onprobe** so as not to bother you with information about the same event. If you want **onprobe** to resume alarm evaluation with exactly the same conditions and sampling frequency, specify the time interval after which the alarm should be reactivated.

Type the desired time interval in the text box and then select a desired time unit from the list box. To reschedule the alarm request immediately, type 0. To trigger the alarm only once (that is, without resuming its evaluation), click **Never**.

Deliver to

In the **Deliver to** group, specify the output method for a triggered alarm. To select an output method, click the desired button and where necessary, type the required text, as follows:

- Click **Popup alarm** to project the alarm and attached message on the screen. **Popup alarm** is enabled in session mode only.
- Click **Mail** and then type one or more electronic mail addresses or aliases.
- Click **Pipe** to send alarm output through the UNIX pipe (for instance, to a program that sends text to a beeper). Type the desired UNIX command.
- Click **Log File** and then type the name of the file. If triggered, alarm output is appended to the end of the file.

If you click **Popup alarm**, the reply is displayed in graphic format in the Alarm Display window. In all other cases, the reply is sent in ASCII format and includes a time stamp, indicating time of receipt.

Examples

The first example shows the reply received after an SMI alarm was triggered:

```
Sun Jul 30 12:54:15 1998
Alarm On SQL Statement "select count (*) from sysessions
having count (*) > 2"
More than one user is currently connected
(count (*)) = 3
```

The second example shows the reply received after a Cockpit alarm was triggered:

```
Sun Jul 30 12:54:15 1998
Alarm On cached_writes With Criteria cached_writes < 0.85
Cached writes percentage is too low
cached_writes = 0.2 (ERROR)
```


Modifying an Existing Alarm

To change the definition of an existing alarm, click the desired alarm in the Alarm Editor and choose **Edit→Modify**. Enter the desired changes in the Alarm Request dialog box. For information about the various alarm components, refer to [“Defining a New Alarm” on page 2-9](#).



Important: If you change a session alarm file to a permanent alarm file, take care to specify an output method other than **Popup alarm** for each alarm in the file. Popup display is enabled in session mode only and is automatically disabled when you switch modes from session to permanent. Any alarm whose only output is popup display has, as a consequence, no output method because no way exists to transmit the information. Switching back to session mode enables the popup option. For more information, refer to [“Switching Between Alarm File Modes” on page 2-8](#).

Deleting an Existing Alarm

To delete an alarm, click the desired alarm in the Alarm Editor and choose **Edit→Delete**.

DB/Cockpit Alarm Syntax

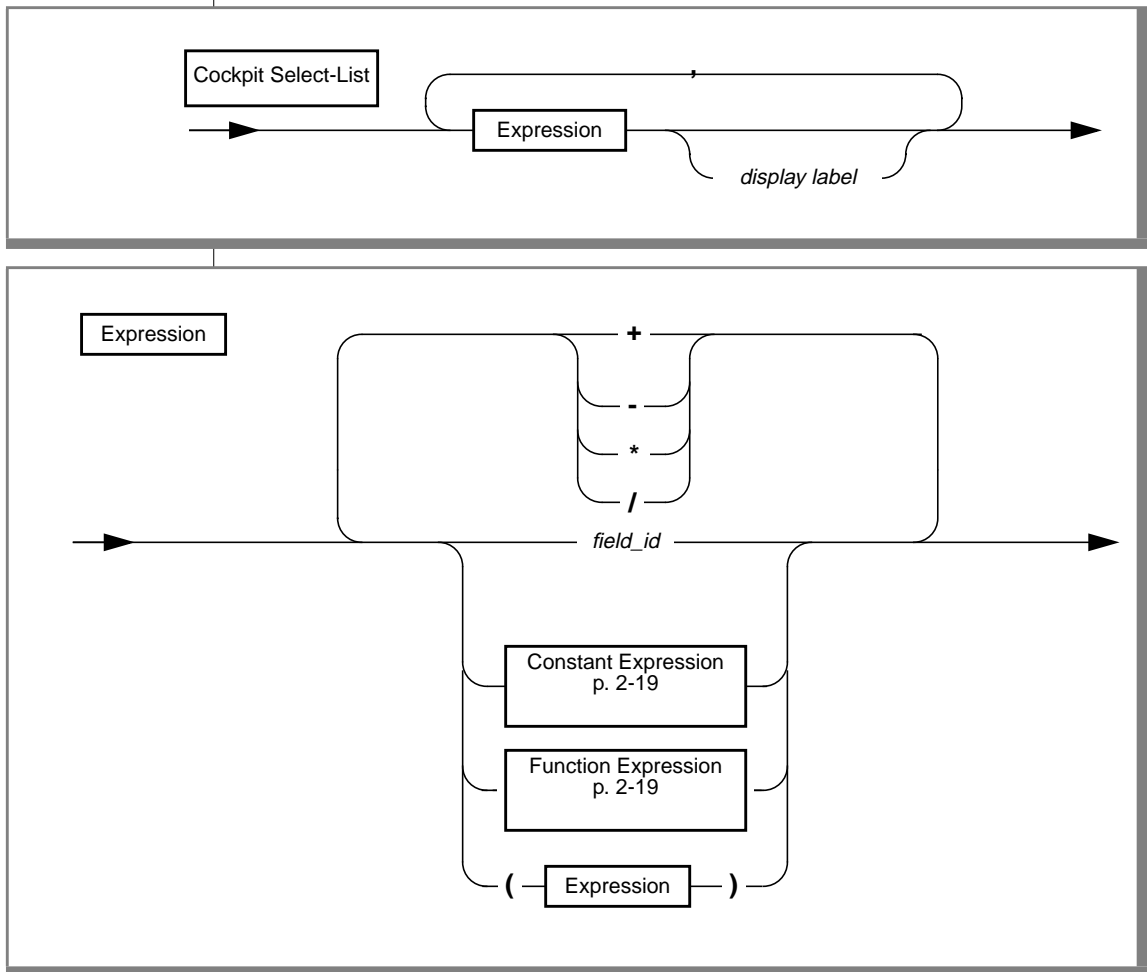
This section describes the following syntax that you use to define Cockpit-type alarms:

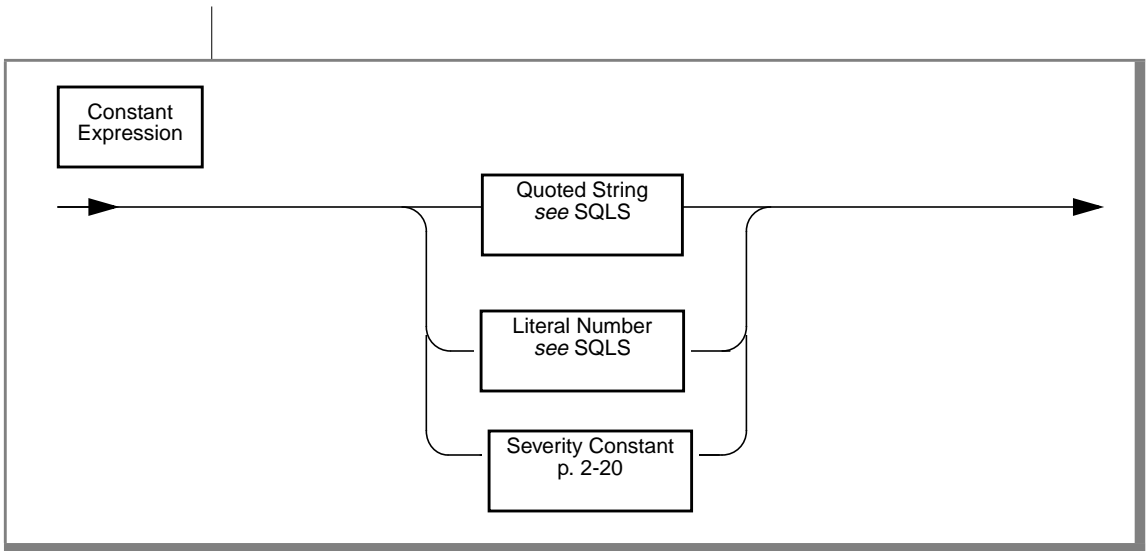
- DB/Cockpit select list syntax
- DB/Cockpit criteria syntax
- Field-IDs

DB/Cockpit Select List Syntax

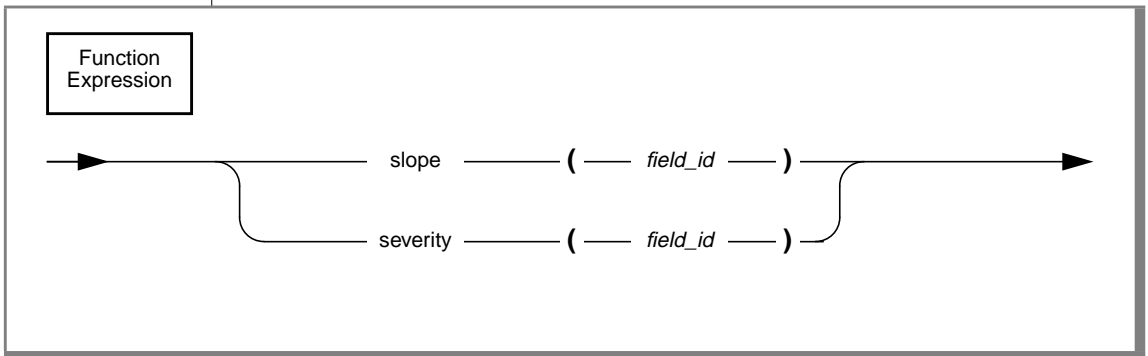
The select list defines the data to be returned when an alarm is triggered. The select list contains one or more field-ID based expressions. If the alarm is triggered, the value of each expression listed is returned.

The syntax of a Cockpit select list resembles the syntax of a SELECT list of an SQL SELECT statement.





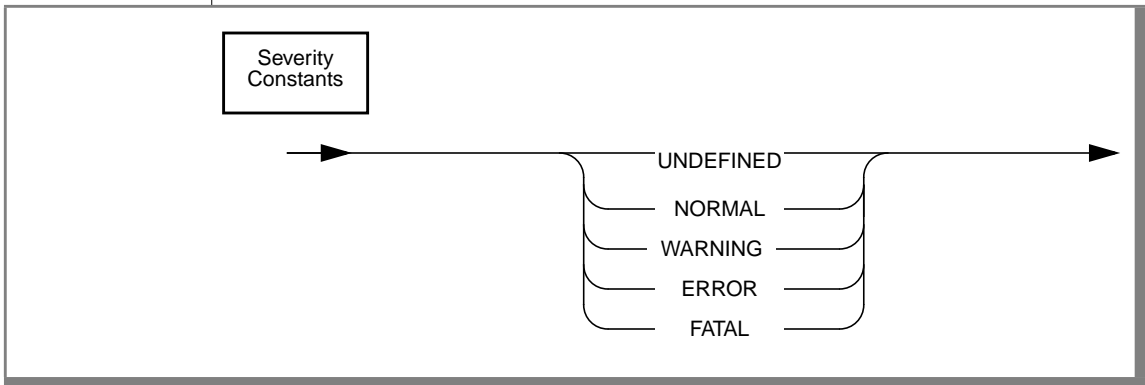
Literal Number can be an integer or a decimal number.



SLOPE and SEVERITY are unique DB/Cockpit functions:

- SLOPE() measures changes in the sampled value between two samples. The value returned is expressed in units per second.
- SEVERITY() returns a severity constant that expresses the current severity value of the sampled field-ID.

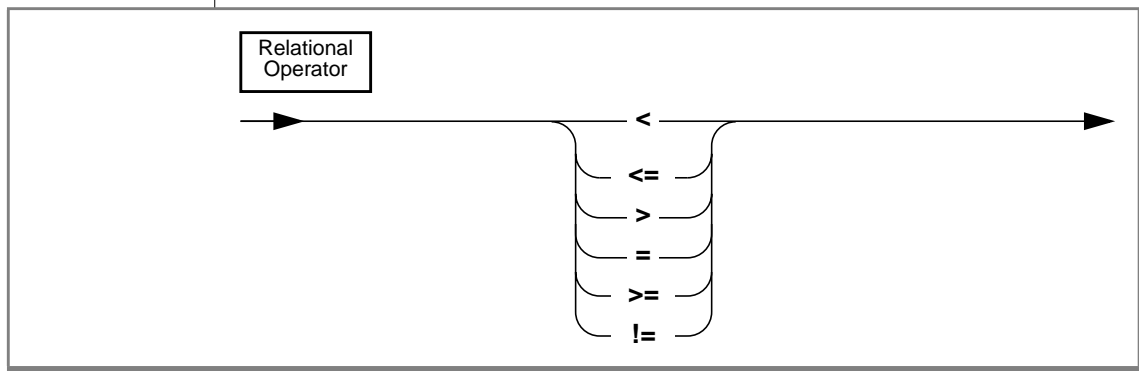
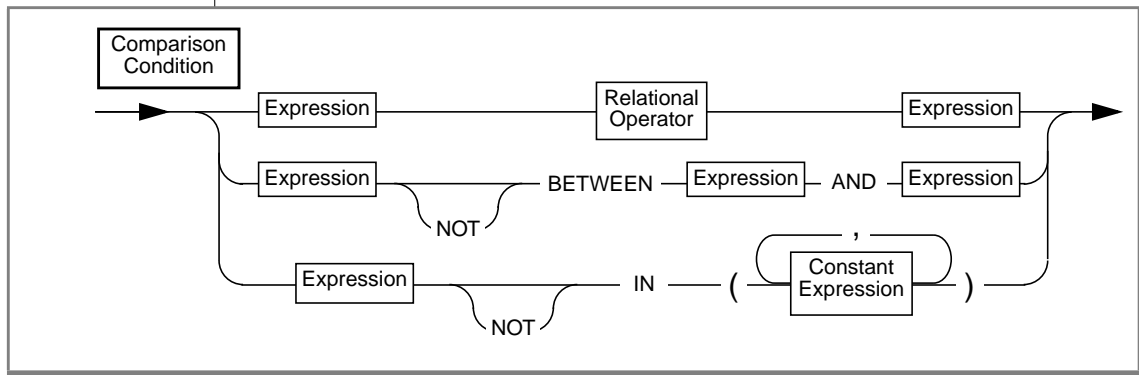
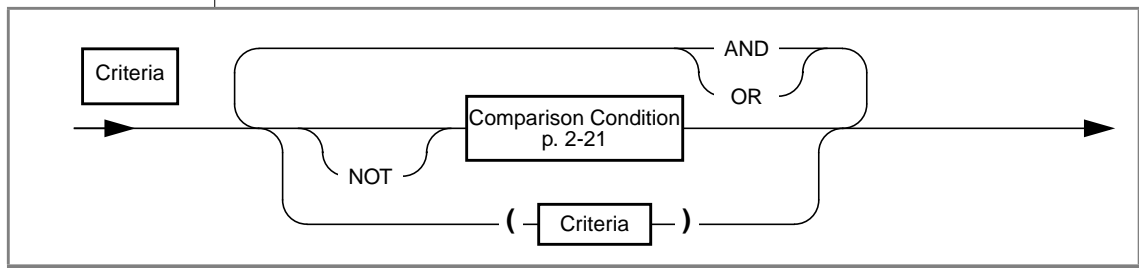
This function can return one of the following values: UNDEFINED, NORMAL, WARNING, ERROR and FATAL. For details about these values, refer to [Chapter 4, “The Severity Analyst.”](#)



DB/Cockpit Criteria Syntax

Cockpit criteria are field-ID based conditions that **onprobe** evaluates. When they prove true, the alarm is triggered.

The syntax of Cockpit criteria resembles the syntax of the WHERE clause of an SQL SELECT statement.



Field-IDs

You can use the following three types of field-IDs:

- Any parameter included in the **sysprofile** table of the **sysmaster** database
For more information, refer to your [Administrator's Guide](#).
Some of the **sysprofile** parameters have aliases whose names resemble the column titles in the **onstat-p** output. For a full list of aliases, refer to the following tables.
- Any Dynamic Server configuration parameter
Take care to enter these field-IDs in uppercase characters. For more information, refer to your [Administrator's Guide](#).
The values of most configuration parameters are retrieved from Dynamic Server reserved pages in order to reflect any changes entered since the last Dynamic Server boot. Several parameters, which are not available in the reserved pages, are retrieved from the **onconfig** configuration file. For more information about reserved pages, refer to your [Administrator's Guide](#).
- Unique DB/Cockpit field-IDs

The following table lists all the DB/Cockpit field-IDs and their aliases where applicable.

Recommended DB/Cockpit Field-ID	Alias DB/Cockpit Field-ID	Description
boot_time		Dynamic Server boot time
cached_reads		[(Number of reads from shared-memory buffers) - (Number of actual reads from disk)] / (Number of reads from shared-memory buffers)*
cached_writes		[(Number of writes to shared-memory buffers) - (Number of actual writes to disk)] / (Number of writes to shared-memory buffers)*
dirty_buffs	dirty_buffs_per	(Number of dirty buffers in shared memory) / (Total number of buffers)

(1 of 2)

Recommended DB/Cockpit Field-ID	Alias DB/Cockpit Field-ID	Description
last_ckpt		Time of last checkpoint
loglogs_used		(Total used space in logical logs) / (Total allocated space in logical logs)
online_mode		Current Dynamic Server operating mode, followed by optional data-replication type of the database server, optional checkpoint status and optional long-transaction indicator
online_uptime		Time passed since Dynamic Server boot
page_size_in_K		Page size in kilobytes
systemcpu		Total system CPU time in seconds that all user threads use*
time_since_ckpt	ckpt_interval	Time passed since last checkpoint
totcpu		Sum of systemcpu and usercpu1
totdiskio		Total number of actual reads from and writes to disk*
totshmem		Total size of Dynamic Server resident, virtual and message shared-memory segments in KB
usercpu		Total user CPU time in seconds that all user threads use*

(2 of 2)

* Since Dynamic Server boot time or last **onstat-z** command.

The following table lists the **sysprofile** field-IDs that have alias DB/Cockpit field-IDs.

sysprofile Field-ID	Alias DB/Cockpit Field-ID	Description
btradata	ixda_RA	Data pages read aheads through leaf [*]
btraidx	idx_RA	Leaf read aheads through parent [*]
dpra	dp_RA	Data pages read aheads [*]
iscommits	iscommit	iscommit calls [*]
isdeletes	isdelete	isdelete calls [*]
isreads	isread	isread calls [*]
isopen	isopens	isopen calls
isrewrites	iswrite	iswrite calls [*]
isrollbacks	isrollback	isrollback calls [*]
isstarts	isstart	isstart calls [*]
iswrites	iswrite	iswrite calls [*]
latchreqs	lchreqs	Number of latch requests [*]
rapgs_used	RA_pgsused	Read-ahead pages used [*]

^{*} Since Dynamic Server boot time or last **onstat-z** command.

Managing Alarm Files

The **File** and **Options** menus in the Alarm Editor contain commands that allow you to manage an alarm file.

Opening an Alarm File

To edit an existing alarm file or modify specific alarms within the file, you must first open the alarm file in the Alarm Editor.

***Tip:** If the permanent alarm file resides on a remote computer and you have no NFS access to the file, remote copy it before opening the file.*

To open an alarm file, choose **File→Open**. When you choose this menu item, a standard file-selection dialog box opens from which you can select the desired alarm file.

***Important:** If necessary, do not forget to change modes (session versus permanent) before you begin working in the Alarm Editor. As mentioned previously, some features are mode dependent.*

If you entered changes in the current alarm file and request to open a new file, you are warned that the current file has not been saved.

Saving an Alarm File

The Alarm Editor provides the following three menu items that allow you to save the changes entered in the current alarm file:

- To save the current alarm file under its original name, choose **File→Save**.
- To save the current file directly to the default **permalrm** or **sessalrm** file in the **oncockpit \$INFORMIXDIR/etc** directory, choose **File→Save As permalrm** or **File→Save As sessalrm**. (The name of this menu item changes to reflect the current file mode.)

If you copied **permalrm** file from a remote computer for editing, do not forget to copy it back.

- To save the current alarm file under another name, choose **File→Save As**. The **Save As** menu item opens a standard file-selection dialog box where you can assign a new name for the alarm file and specify its location.





Important: Saving alarms does not update the active alarm file (neither permanent nor session) loaded in **onprobe**. To override the active session alarm file, you must use the **Options→Replace** menu item, as explained in the next section. To override the active permanent alarm file, you must restart **onprobe**.

Replacing the Active Session Alarm File

You can replace the active session alarm file immediately during run time. To do so, open the session alarm file that you want to activate in the Alarm Editor and choose **Options→Replace**.

Replacing the active permanent alarm file requires another approach. Although you can edit permanent alarms from the Alarm Editor, you cannot replace the active permanent alarm file with your updates. For more information, refer to [“Activating Alarm Files” on page 2-26](#).

Activating Alarm Files

By default, **onprobe** loads (that is, activates) the permanent alarm file named **permalrm** in **\$INFORMIXDIR/etc**.

By default, **oncockpit** loads the session alarm file named **sessalrm** in **\$INFORMIXDIR/etc**.

The **onprobe** and **oncockpit** programs can also load any other alarm file, provided that it is specified at start-up with the **-alarms** parameter in the command line.



Important: If you use the default alarm files, **permalrm** and **sessalrm**, and **onprobe** and **oncockpit** run on different computers, their corresponding **\$INFORMIXDIR** directories can reside in different file systems.



DB/Cockpit comes ready-made with the following four template alarm files:

- Two identical permanent alarm files, named **permalrm** and **permalrm.std**, reside in the **onprobe \$INFORMIXDIR/etc** directory and contain a suggested set of permanent alarms.
- Two identical session alarm files, named **sessalrm** and **sessalrm.std**, reside in the **oncockpit \$INFORMIXDIR/etc** directory and contain a suggested set of session alarms.

*Tip: DB/Cockpit offers the **permalrm.std** and **sessalrm.std** files as copies for reference.*

Preparing the Template Alarm Files for Activation

The **permalrm** file and the **sessalrm** file contain an identical set of alarms. The **Mail**, **Pipe**, and **Log File** options in the **Deliver To** group are disabled for all alarms because they are site dependent. Although the two files contain an identical set of alarms, the following two major differences exist between the files:

- The alarms in **permalrm** are inactive, while those in **sessalrm** are active.
- The **Popup Alarm** output method is enabled for all alarms in the **sessalrm** file. (As already mentioned, **Popup Alarm** is irrelevant for permanent alarms.)

You can easily edit both these alarm files (as well as any other alarm file) from the **oncockpit** Alarm Editor.

To prepare the **permalrm** file

1. Choose **File→Open** to open **\$INFORMIXDIR/etc/permalrm** in the Alarm Editor.
If the **permalrm** file resides on a remote computer and you have no NFS access to the file, remote copy it before you open the file.
2. Click the desired alarm in the Alarm Editor and choose **Edit→Modify** to open the alarm that you want to activate in the Alarm Request dialog box.
3. Click **Active** in the **Type and Field-IDs** group.

4. Select the desired output method in the **Deliver To** group and modify its contents if necessary.
5. If necessary, customize other values in the alarm definition according to the parameters of your database server.
6. After you modify the desired alarms, choose **File→Save As permalarm** to save the current file under \$INFORMIXDIR/etc.

If you copied the **permalarm** file from a remote computer, remember to copy it back.

To prepare the sessalarm file

1. Choose **File→Open** to open \$INFORMIXDIR/etc/sessalarm in the Alarm Editor.
2. Click the desired alarm in the Alarm Editor and choose **Edit→Modify** to open the alarm that you want to activate or customize.
3. To redirect alarm results to another destination or to more than one destination, select the desired output method in the **Deliver To** group and modify its contents if necessary.
By default, **Popup alarm** is selected.
4. If necessary, customize other values in the alarm definition according to the parameters of your database server.
5. After you modify the desired alarms, choose **File→Save As sessalarm** to save the current file.

Activating the Default Alarm Files

If you modified alarms in the **permalrm** file, you must restart **onprobe** for changes to take effect.

Unlike **permalrm**, you can activate the modified **sessalrm** file directly from the Alarm Editor in **oncockpit** by choosing **Options→Replace**.

Activating Other Alarm Files

You can create as many permanent and session alarm files as you want and switch between the currently active files and other files.

The easiest way to activate another alarm file (if it is not the default) is to specify its name at start-up with the **-alarms** command-line option. The **-alarms** parameter is relevant for both **onprobe** and **oncockpit**.

The following example shows how to activate a permanent alarm file named **permalrm.gen**:

```
onprobe -alarms permalrm.gen
```

The following example shows how to activate a session alarm file named **sessalrm.tom**:

```
oncockpit -alarms sessalrm.tom
```



***Tip:** Alternatively, if you do not want to use the **-alarms** parameter, you can save the current **permalrm** or **sessalrm** file under another name (if you want to use the file in the future) and rename the alarm file you want to activate to the suitable default name.*

The Alarm Display Window


Because **oncockpit** operates in a graphic environment, you can request to display results from triggered session alarms on the screen. When a session alarm is triggered, the Alarm Display dialog box opens and shows alarm output.

To enable this option, click **Popup alarm** in the Alarm Request dialog box. [Figure 2-5](#) shows the Alarm Display dialog box for an SMI alarm.

Figure 2-5

The Alarm Display Dialog Box That Shows SMI Alarm Output

DB/Cockpit - Alarm Display (pern_shm)

Type and Field IDs: 

Message:

Alarm Data:

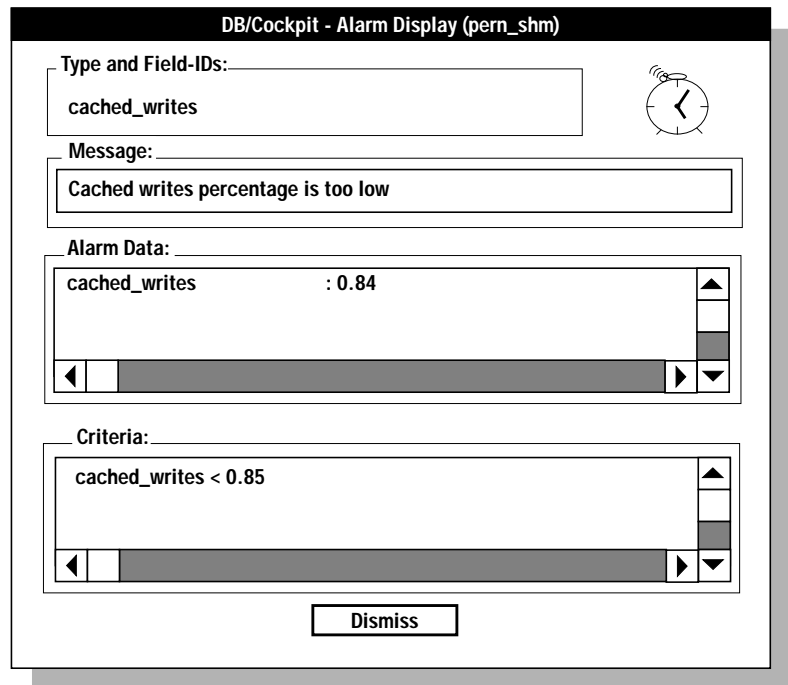
sid	: 444	▲
username	: root	▼

SQL text:

```
select * from syssessions where
username='root'
```

Figure 2-6 shows the Alarm Display dialog box for a **Cockpit** alarm.

Figure 2-6
The Alarm Display Dialog Box That Shows Cockpit Alarm Output



To close the Alarm Display dialog box, click **Dismiss**.

After the alarm is triggered, it is rescheduled according to the value specified in the Alarm Request dialog box, irrespective of whether its popup display was dismissed or not.

The DB/Cockpit Viewer

In This Chapter	3-3
Using the Drill-Down Method	3-4
Viewing Long Identifiers	3-4
Refreshing Information in Views	3-5
The DB/Cockpit Main Window	3-5
The Spaces View.	3-8
The Space Information View	3-10
The Chunk Spaces View.	3-12
The Chunk Space Information View	3-13
The Chunk Disk Fragmentation View	3-15
The Databases View	3-18
The Database Information View	3-20
The Tables View	3-22
The Table Profile View	3-24
The Table Information View	3-26
The Table Disk Fragmentation View	3-28
The Sessions View	3-30
The Session Profile View	3-32
The SQL Statement View	3-35

The Logical Logs View	3-37
The Physical Log View	3-39
The Data Replication View	3-41
The Virtual Processors View	3-43
The Active VPs View	3-45
The VP Information View	3-46
The I/O VP Queues View	3-49
The Shared Memory View	3-51
The SHM Buffers View	3-54
The SHM Segments View	3-56
The ISAM View	3-58
The Overflow View	3-59
The Waits View	3-60
The Disk View	3-62

In This Chapter

The DB/Cockpit tool provides a variety of *views* or windows that open into the database server to allow you to focus on its values according to the subject that you want to monitor.

Some of the views handle static information while others are dynamic in nature. The **Spaces** view, which shows the physical structure of data, or the **Databases** view, which shows the logical structure of data, are static views. The **Sessions** view, which monitors user access to database tables, or the **Shared Memory** view are dynamic views and the data they display changes.

Depending on the subject, views can present the data in text and graphic format. Some views present the data in both formats, allowing you to decide whether a quick glance and a general idea of matters are enough or whether you need accurate values.

DB/Cockpit supports views about dbspaces, databases, sessions, logical and physical logs, data replication, virtual processors, and shared memory.

***Tip:** Each view presented in this chapter includes a screen example and a table that details the field-IDs included in the view; where necessary, additional remarks are added. The field-IDs listed in these tables are used in severity expressions.*



Using the Drill-Down Method

Most views make use of the *drill-down* method. You begin at the top view, where general information about a certain subject is displayed, and from this view you *drill down* to specific information, presented in *subviews*. This method relies on buttons that enable you to navigate between several related views and access additional information from the main (topmost) view. Usually, the main view presents the most important or general information, while the drill-down views hold detailed and specific information.

Several views, such as **Spaces** or **Databases**, present a scrollable list of items. Initially, the first item in the list is selected by default. To select another item, click it. To explore other views for further information, select the item that interests you and click the relevant drill-down button. All buttons are located at the bottom of the views.

Viewing Long Identifiers

When an identifier ends with three dots (...), this identifier is too long and it cannot be displayed completely in this particular view. To display the identifier, select it with the mouse without clicking. A popup window opens and displays the entire identifier.

In some views when an identifier is too long, it is truncated. In most of these views, you can use the drill-down method to see the entire identifier.

The subviews are scrollable windows so you might not have a problem displaying the full name of any identifier. For example, in the **Databases** view, the top level lists all the databases in the Dynamic Server system. If a database name is longer than 33 characters, it will truncate.

To receive general information about a specific database, including its full name, click the **DB Info** button using the drill-down method. Use this method for the **Tables** view, **Sessions** view, and **Spaces** view.

Refreshing Information in Views

DB/Cockpit views provide the following manual and automatic refresh methods:

- **Manual refresh.** You must click the suitable refresh button in the view to initiate a new display of the information. DB/Cockpit offers the following two refresh buttons:

- **Refresh**
- **Apply**

The main purpose of these buttons is to retrieve information according to new display criteria. To refresh a view, click **Apply** without first changing the current criteria.

The **Chunk Fragmentation** view is the one exception. Although this view belongs to the manual refresh method, to refresh the data display for the current (selected) chunk, you have to reselect it by clicking.

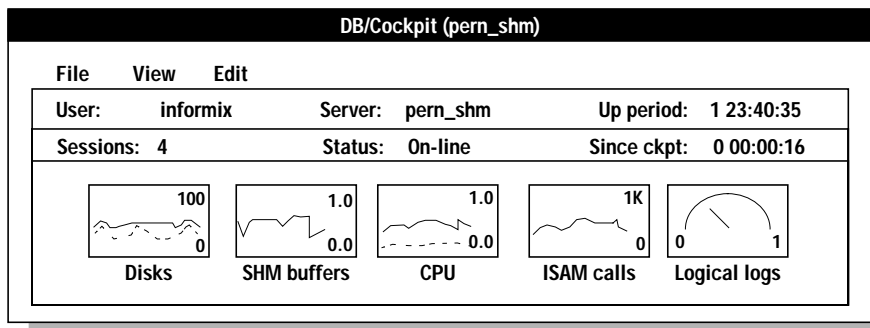
- **Automatic refresh.** Any view that does not provide one of the above refresh methods, refreshes automatically.

The DB/Cockpit Main Window

When you launch **oncockpit**, the DB/Cockpit main window is displayed. It monitors the main Dynamic Server system activities. As opposed to other views that this chapter describes, the DB/Cockpit main window includes a menu bar that allows you to access other views and perform various DB/Cockpit functions.

Most of the field-IDs in the DB/Cockpit main window are retrieved directly from shared memory to reduce CPU overhead.

Figure 3-1
The DB/Cockpit Main Window



Important: The DB/Cockpit main window refreshes automatically.

Any field-ID checked in the **Field-ID** column and all field-IDs in the **Remarks** column in the following table are in one of the following categories:

- Parameters included in the **sysprofile** table of the **sysmaster** database
- Parameters that appear in the **onconfig** configuration file
- Unique DB/Cockpit field-IDs

For more information, refer to [“Field-IDs” on page 2-22](#).

The **Remarks** column shows the formulas that DB/Cockpit uses to calculate values. Slope() is a unique DB/Cockpit function that measures changes in the sampled value over a period of time.

The fields in the lower section of the DB/Cockpit main window are depicted as graphic displays. In the following table, (1) and (2) represent the upper and lower lines of the graphs, respectively.

The DB/Cockpit main window includes the following fields.

Label	Field-ID	Remarks
User	username	User that launched onprobe
Sessions	num_sessions	Number of sessions currently connected
Server	INFORMIXSERVER	The INFORMIXSERVER environment variable used to launch onprobe
Status	√online_mode	
Up period	√online_uptime	
Since ckpt	√ckpt_interval	Time of last checkpoint
Disks	(1) (no field-ID)	slope (totdiskio)
	(2) (no field-ID)	slope (dskwrites)
SHM buffers	(no field-ID)	dirty_buffs_per
CPU	(1) (no field-ID)	slope (totcpu)
	(2) (no field-ID)	slope (systemcpu)
ISAM calls	(no field-ID)	slope (isamtot)
Logical logs	loglogs.used	Graphic display gauging values between 0 and 1



The Spaces View

The **Spaces** view deals with data storage, focusing on the physical structure, and use of Dynamic Server spaces.

Important: *Chunk Disk Fragmentation for blobspaces and sbspaces has not been implemented. Therefore, the **Disk Frag** button in **Spaces** list is not enabled when you select a blobspace or an sbpace.*

The main **Spaces** view provides general information about dbspaces.

Figure 3-2
The Spaces View

DB/Cockpit - Spaces (pern_shm)								
dbspaces list								
space name	correct	blob	0%	50%	100%	total(M)	free(M)	mirror
rootdbs	y		<div><div></div></div>			21	29	
db1	y		<div><div></div></div>			2	2	✓
blobsp1	y		<div><div></div></div>			2	2	✓
sblobsp1	y	simple	<div><div></div></div>			2	2	✓
db2	n	smart	<div><div></div></div>			2	2	

Close

Space Info

Chunks

Disk Frag

Refresh

The main **Spaces** view includes the following fields.

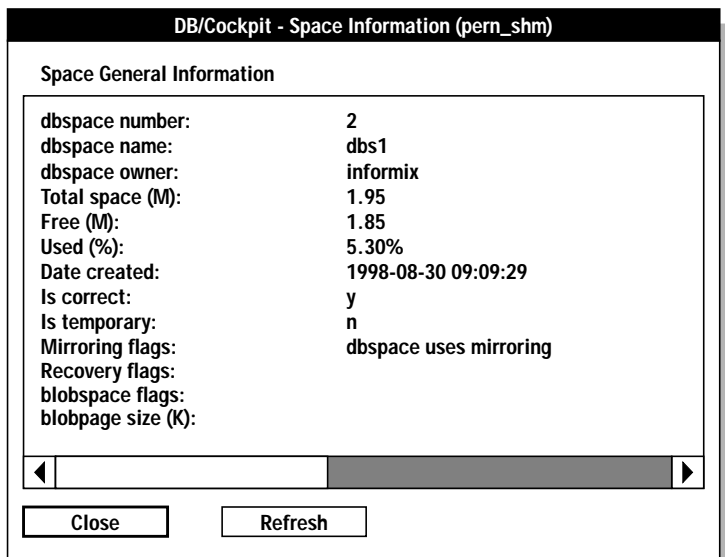
Label	Field-ID	Remarks
space name	spaceslist.name	
correct	spaceslist.correct	Accepts Y/N values
blob	spaceslist.blob	Accepts simple/smart values
0% 50% 100%	spaceslist.used	Graphic display gauging values between 0 and 1
total (M)	spaceslist.total	In MB, rounded
free (M)	spaceslist.free	In MB, rounded
mirror	spaceslist.mirror	Accepts √/blank values

The Space Information View

To display additional information about a dbspace, select a dbspace from the main **Spaces** view and drill down with the **Space Info** button.

Figure 3-3

The Space Information View



The **Space** Information view includes the following fields.

Label	Field-ID	Remarks
dbspace number	spaceinfo.dbspace_no	
dbspace name	spaceinfo.name	
dbspace owner	spaceinfo.owner	
Total space (M)	spaceinfo.total	In MB
Free (M)	spaceinfo.free	In MB
Used (%)	spaceinfo.used	In percent
Date created	spaceinfo.created	Date and time
Is correct	spaceinfo.correct	Accepts Y/N values
Is temporary	spaceinfo.temporary	Accepts Y/N values
Mirroring flags	spaceinfo.mirror	Text field
blobspace flags	spaceinfo.blob	Text field
Recovery flags	spaceinfo.recovery	Text field

The Chunk Spaces View

To display a list of existing chunks of that dbspace, select a dbspace from the main **Spaces** view and drill down with the **Chunks** button.

Figure 3-4
The Chunk Spaces View

DB/Cockpit - Chunk Spaces (pern_shm)

space name: blbsp1

blob:

mirror: ☐

no.	device name	0%	50%	100%	total(M)	free(M)	correct	disk read	disk write
2	blbsp1	<div><div></div></div>			1	1	y	0	3

Close

Chunk Info

Refresh

The **Chunk Spaces** view includes the following fields.

Label	Field-ID	Remarks
no.	chunklist.number	
device name	chunklist.name	Basename
0% 50% 100%	chunklist.used	Graphic display gauging values between 0 and 1
total (M)	chunklist.total	In MB, rounded
free (M)	chunklist.free	In MB, rounded
correct	chunklist.correct	Accepts Y/N values
disk read	chunklist.read_ops	Number of disk reads
disk write	chunklist.write_ops	Number of disk writes

The Chunk Space Information View

To display the following detailed information about the selected chunk, select a chunk from the list in the **Chunk Spaces** view and drill down with the **Chunk Info** button:

- General information
- Blobspace information if the chunk belongs to a blobspace
- Sbspace information if the chunk belongs to an Sbspace
- Mirror chunk information if the chunk is mirrored.

DB/Cockpit - Chunk Space Information (pern_shm)

Chunk General Information

dbspace:	blbsp2
Chunk number:	3
Device name:	/work3/SLAVA_9.20.LATEST_45/area
Device Type:	cooked file
Offset (M):	0.00
Total Space (M):	0.98
Free Space (M):	0.97
Used Space (%):	0.60%
Disk read:	4
Disk write:	3
Is blobspace:	y
Is mirrored:	n
Status flags:	chunk is online

◀
▶

Blobspace Information ☒

Page size (K):	2
Total pages:	500
Free pages:	497

Sbspace Information ☐

Total pages:	
Free pages:	
Total data pages:	

Mirror Chunk Information ☐

Device name:	
Offset (M):	
Disk read:	
Disk write:	
Status flags:	

Close

Refresh

Figure 3-5
The Chunk Space
Information View

The **Chunk Space Information** view includes the following fields.

Label	Field-ID	Remarks
dbspace	chunkinfo.dbspace	
Chunk number	chunkinfo.number	
Device name	chunkinfo.device	Full path
Device Type	chunkinfo.dev_type	Values accepted: raw device, cooked file
Offset (M)	chunkinfo.offset	In MB
Total Space (M)	chunkinfo.total	In MB
Free Space (M)	chunkinfo.free	In MB
Used Space (%)	chunkinfo.used	In percent
Disk read	chunkinfo.read_ops	Number of disk reads
Disk write	chunkinfo.write_ops	Number of disk writes
Is blobspace	chunkinfo.blob	Accepts Y/N values
Is mirrored	chunkinfo.mirror	Accepts Y/N values
Status flags	chunkinfo.status	Text field
Page size (K)	chunkinfo.blobpg_size	In KB
Total pages	chunkinfo.sb_totpgs	
Free pages	chunkinfo.sb_freepgs	
Total pages	chunkinfo.totpgs	
Free pages	chunkinfo.freepgs	
Total data pages	chunkinfo.sb_totdpgs	
Device name	chunkinfo.mdevice	Full path
Offset (M)	chunkinfo.moffset	In MB

(1 of 2)

Label	Field-ID	Remarks
Disk read	chunkinfo.mread_ops	Number of disk reads
Disk write	chunkinfo.mwrite_ops	Number of disk writes
Status flags	chunkinfo.mstatus	Text field

(2 of 2)

The Chunk Disk Fragmentation View

To display a graphic presentation of dbspace fragmentation, select a dbspace from the main **Spaces** view and drill down with the **Disk Frag** button. The **Chunk Disk Fragmentation** view allows you to evaluate the amount of free space, occupied space, and fragmented areas at a glance.

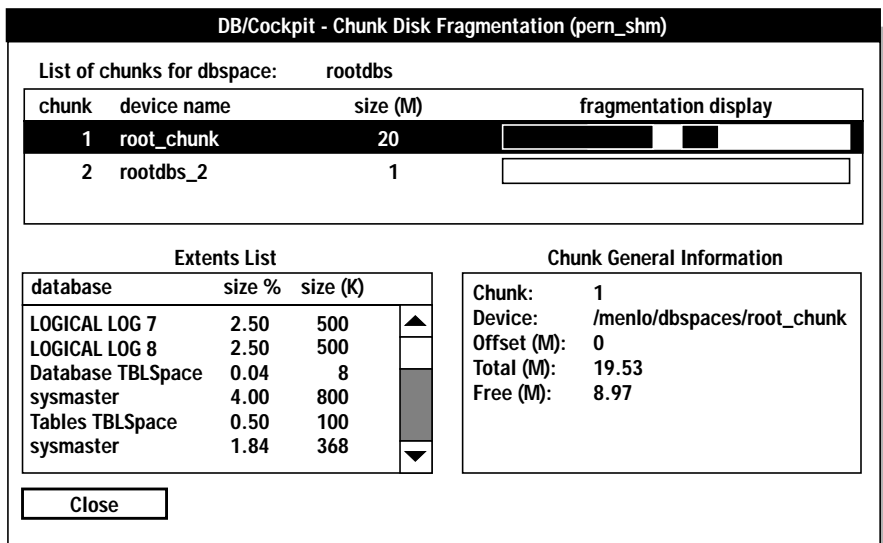
*Tip: The **Disk Frag** button is disabled when blobspaces are selected.*

The upper part of the view shows a list of chunks and a graphic presentation of their fragmentation. Initially, information about the first chunk is shown because it is selected by default. To see detailed information about a specific chunk, select it from the list.



The detailed information about extents in the current chunk as well as general information about the chunk are displayed in the two lower areas of the view.

Figure 3-6
The Chunk Disk Fragmentation View



Important: To refresh the view for the current chunk, click the chunk again.

The **Chunk Fragmentation** view includes the following fields.

Label	Field-ID	Remarks
chunk	dbf_fraglst.number	
device name	dbf_fraglst.device	Basename
size (M)	dbf_fraglst.chunk_size	In MB, rounded
fragmentation display	(no field-ID)	Graphic display that presents dbspace distribution of extents per chunk
database	chk_fraglst.database	
size (%)	chk_fraglst.persize	Extent size, percentage from the total chunk size
size (K)	chk_fraglst.size	Extent size, in KB
Chunk	chunkinfo.number	
Device	chunkinfo.device	Full path
Offset (M)	chunkinfo.offset	In MB
Total (M)	chunkinfo.total	Chunk size, in MB
Free (M)	chunkinfo.free	In MB

The Databases View

The **Databases** views enable you to investigate the logical structures within the Dynamic Server system. The main, top-most view lists all the databases in the Dynamic Server system, sorted by the number of sessions that are currently connected.

Figure 3-7
The Databases View

DB/Cockpit - Databases (pern_shm)				
Databases List				
database	log status	dbspace	tbl no.	sess con.
db1	unbuffered	s1addmir	30	2
sysmaster	unbuffered	rootdbs	56	2
sports	unbuffered	rootdbs	40	1
stores_demo	no logging	rootdbs	39	0
et1	no logging	et1_dbspace	35	0
ecotoolsdb	no logging	rootdbs	32	0
syspg4gl	no logging	rootdbs	32	0
elitest	buffered	rootdbs	31	0
restock	no logging	rootdbs	30	0
sarit	unbuffered	dummy12	30	0

Close
DB Info
Tables
Refresh

The **Databases** view includes the following fields.

Label	Field-ID	Remarks
database	databaselst.database	
log status	databaselst.log	Values accepted: no logging, unbuffered, buffered, ansi
dbspace	databaselst.dbspace	The dbspace where the database was created
tbl no.	databaselst.tables_no	Number of tables. Counts only tables that have extents on the disk, for example, regular , system , and temporary tables. Synonyms, views, VERSION tables, and pseudo tables are not counted here.
sess. con.	databaselst.sess_no	Number of sessions currently connected to the database

The Database Information View

From the main **Databases** view, you can select a database and drill down with the **DB Info** button to display general information about a specific database.

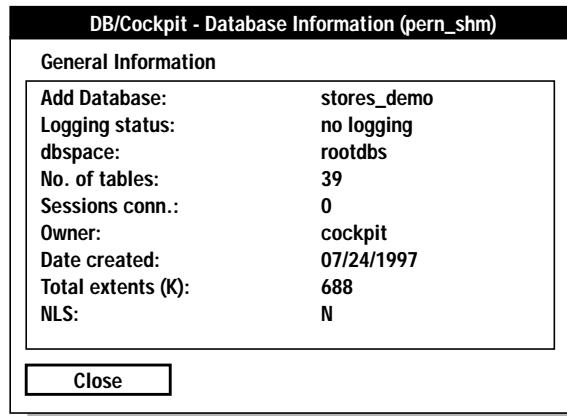


Figure 3-8
*The Database
Information View*



Important: The **Database Information** view refreshes automatically.

The **Database Information** view includes the following fields.

Label	Field-ID	Remarks
Add Database	dbinfo.database	
Logging status	dbinfo.log	Values accepted: no logging, unbuffered, buffered, ansi
dbspace	dbinfo.dbspace	dbspace where the database was created
No. of tables	dbinfo.tables_no	
Sessions conn.	dbinfo.sess_no	Number of sessions currently connected to the database
Owner	dbinfo.owner	
Date created	dbinfo.created	
Total extents (K)	dbinfo.totext	In KB
NLS	dbinfo.isnls	Accepts Y/N values

The Tables View

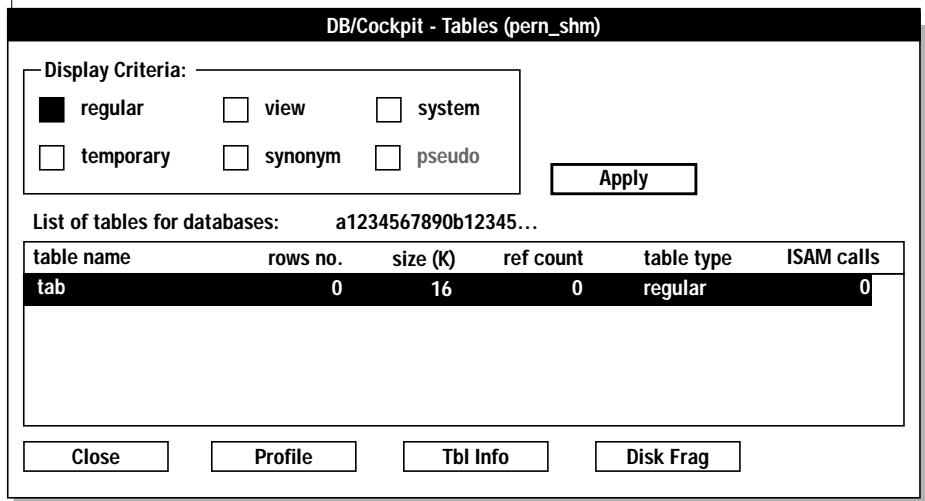
Another view that you can open is the **Tables** view. To see a list of tables created in a certain database, select the desired database in the main **Databases** view and click **Tables**. Tables in this view are sorted alphabetically.

The upper part of the view, named Display Criteria, enables you to define criteria for retrieving tables. DB/Cockpit recognizes several table types:

- **system**—system catalog table
- **temporary**
- **view**
- **regular**
- **synonym**
- **pseudo**—pseudo table that Dynamic Server maintains in the **sysmaster** (SMI) database only

The **temporary** button in the Display Criteria area represents **temporary** and **tmp-index** tables. The **regular** button represents **regular** and **index** tables. By default, **regular** is selected.

Figure 3-9
The Tables View



The **Tables** view includes the following fields.

Label	Field-ID	Remarks
Display Criteria	tablelist.tabtype	See explanations at the beginning of this section
table name	tableslist.tabname	
rows no.	tableslist.rowsno	
size	tableslist.totext	Total size of table extents, in KB
ref count	tableslist.usercnt	Number of users currently accessing the table
table type	tableslist.tabtype	See explanations at the beginning of this section
ISAM calls	tableslist.isamtot	

Note that for some table types, part of the table parameters might not be displayed. For example, tables of type **pseudo**, **view**, **synonym**, and tables named “VERSION” have no extents on the disk; therefore all the disk-related parameters (size, Total extents size, First extent size, and so forth) are empty. The current implementation is capable of presenting **ref count** only for tables that have extents on the disk. For **pseudo**, **view**, **synonym**, and tables named “VERSION”, this parameter in Tables List is empty. Other unavailable or nonrelevant parameters in Tables List and Table Info are shown empty.

The Table Profile View

You can select a table name from the list in the **Tables** view and drill down with the **Profile** button to display profiling information about the selected table, as [Figure 3-10](#) shows.

Important: Profiling information is not available for tables of type **synonym**, **view**, and **pseudo** because they have no extents on the disk. When you select a table of one of these types in the **Tables** view, the **Profile** button is grayed.

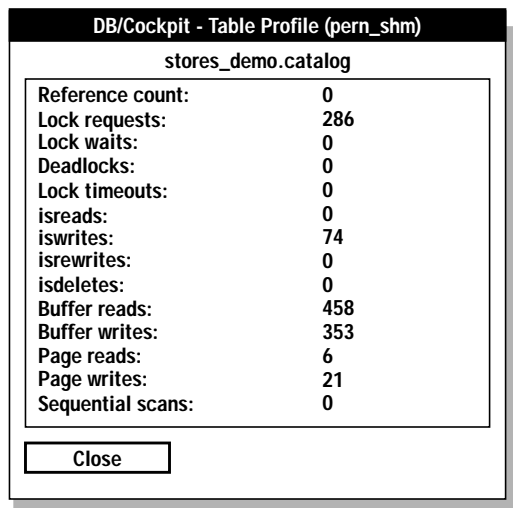


Figure 3-10
The Table Profile
View

Important: The **Table Profile** view refreshes automatically.

The **Table Profile** view includes the following fields.

Label	Field-ID	Remarks
Reference count	tabprof.usercnt	Number of current references to the table
Lock requests	tabprof.lockreqs	
Lock waits	tabprof.lockwts	
Deadlocks	tabprof.deadlks	
Lock timeouts	tabprof.lktouts	
isreads	tabprof.isreads	
iswrites	tabprof.iswrites	
isrewrites	tabprof.isrewrites	
isdeletes	tabprof.isdeletes	
Buffer reads	tabprof.bufreads	
Buffer writes	tabprof.bufwrites	
Page reads	tabprof.pagreads	
Page writes	tabprof.pagwrites	
Sequential scans	tabprof.seqscans	

The Table Information View

In addition to profiling information about a certain table, you can also receive general information about it. To receive the **Table Information** view, select the desired table in the **Tables** view and click **Table Info**.

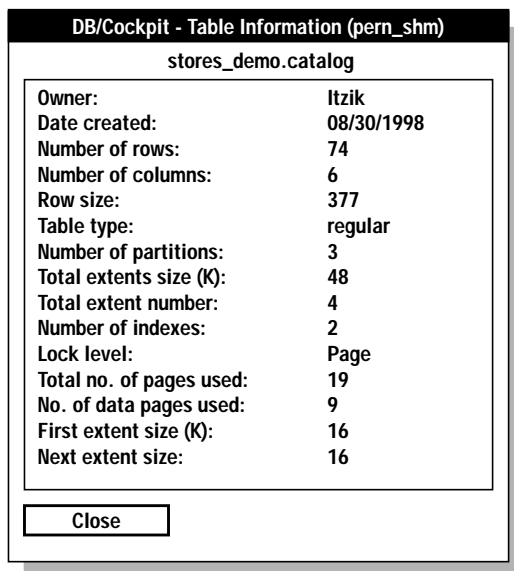


Figure 3-11
The Table
Information View



Important: The **Table Information** view refreshes automatically.

The **Table Information** view includes the following fields.

Label	Field-ID	Remarks
Owner	tabinfo.owner	
Date created	tabinfo.created	
Number of rows	tabinfo.rows_no	
Number of columns	tabinfo.cols_no	
Row size	tabinfo.rowsize	In bytes
Table type	tabinfo.tabtype	For information on table types, see explanations in “The Tables View” on page 3-22
Number of partitions	tabinfo.part_no	In KB
Total extents size	tabinfo.totext	In KB
Total extent number	tabinfo.ext_no	Number of extents
Number of indexes	tabinfo.idx_no	
Lock level	tabinfo.locklevel	Values accepted: Row, Page
Total no. of pages used	tabinfo.totpgsused	
No. of data pages used	tabinfo.datapgsused	
First extent size (K)	tabinfo.fextsize	In KB
Next extent size (K)	tabinfo.nextsize	In KB

The Table Disk Fragmentation View

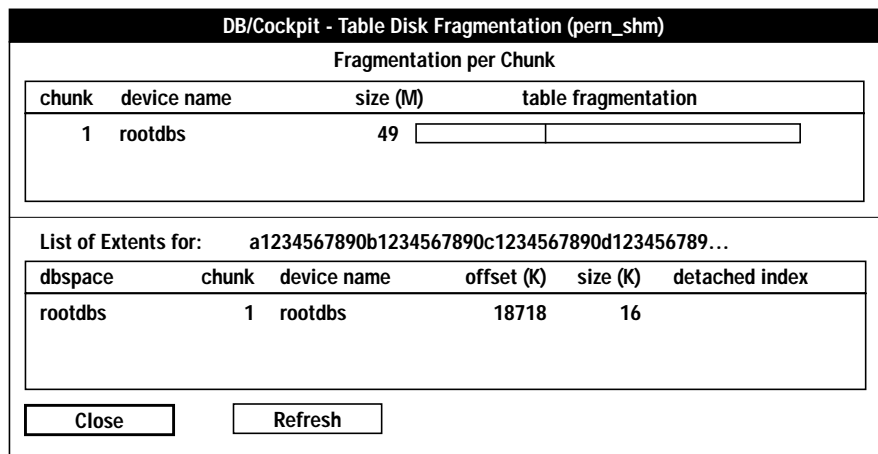
From the list in the **Tables** view, you can access the **Table Disk Fragmentation** view by clicking **Disk Frag**. The **Table Disk Fragmentation** view includes two parts:

- A graphic presentation of fragmentation per chunk
- A list of table extents

Important: When you select a table whose type is **synonym**, **view**, or **pseudo** in the **Tables** view, the **Disk Frag** button is grayed because these tables have no extents on the disk.



Figure 3-12
The Table Disk Fragmentation View



The **Table Disk Fragmentation** view includes the following fields.

Label	Field-ID	Remarks
chunk	extentslist.chunk_no	
device name	extentslist.device	Chunk basename
size (M)	extentslist.chunk_size	Chunk size in MB
table fragmentation	(no field-ID)	Graphic display that presents the distribution of table extents per chunk
dbspace	extentslist.dbspace	
chunk	extentslist.chunk_no	
device name	extentslist.device	Chunk basename
offset (K)	extentslist.offset	Table extent offset in chunk, measured in KB
size (K)	extentslist.extsize	Extent size in KB
detached index	extentslist.det_index	No. of detached indexes

Detached indexes treatment in Tables List, Table Profile, Table Info, and Table Disk Fragmentation has changed. Detached indexes are not recognized as separate table types (**index** and **tmp-index** in the previous implementation). Instead, detached indexes are treated as table partitions. All detached index statistics are included in the corresponding statistics at the table level.

The Sessions View

The **Sessions** view enables you to monitor user activity in the Dynamic Server system. Because the **session** list can reach unmanageable size, Display Criteria are provided in the upper part of the view, which enables you to retrieve and display a desired set of sessions in sorted order.

To set the type of sessions in the list, specify one or more criteria in the **user**, **host**, and **database** text boxes. Wildcards are not supported, so take care to enter full names only.

To sort the list of sessions retrieved, choose a sort key by clicking the **sort by** option box. Available sort keys are assigned an asterisk (*) in the table on [page 3-33](#).

After you define the criteria, click **Apply**.

Important: To refresh the **session** list using the same criteria, click **Apply** again.



Figure 3-13
The Sessions View

DB/Cockpit - Sessions (pern_shm)

Display Criteria: _____

user:

host:

database:

sort by: ▼

Sessions List According to Criteria

sid	user	host	database	pid	connect time	long tx
456	light	pern	stores_demo	6361	0 00:03:59	0
448	vitaly	pern	db1	6278	0 00:22:33	0
454	roy	pern	syspg4gl	6360	0 00:04:22	0
452	mcollins	pern	sports	6359	0 00:04:42	0
420	informix	pern	sysmaster	6184	0 01:19:51	0
444	root	pern	db1	6200	0 01:02:40	0
460	alex	pern	db1	6393	0 00:03:02	0
458	gabi	pern	stores_demo	6377	0 00:03:29	0

The **Sessions** view includes the following fields.

Label	Field-ID	Remarks
sid	seslist.sid	Dynamic Server session ID
user	seslist.user	
host	seslist.host	
database	seslist.db	Database accessed
pid	seslist.pid	UNIX process ID of the client process
connect time	seslist.conntime	Connection duration
(flexible sort field)	seslist.sortfield	Variable field; can display any field in the Session Profile marked by *

The Session Profile View

To drill down to display the **Session Profile** view, select a session from the list in the main **Sessions** view and click **Profile**.

DB/Cockpit - Session Profile (pern_shm)

Profile Information

Sid:	456
User:	light
Host:	pern
Database:	stores_demo
Pid:	6361
tty:	/dev/tty7
Connect time:	0 00:09:48
Total memory (byte):	40960
Used memory (byte):	31400
Locks requested:	69
Locks wait:	0
Locks held:	1
Locks timeout:	0
Deadlocks detected:	0
CPU time (sec):	
No. of long tx:	0
Reads:	32
Writes:	0
Rewrites:	0
Deletes:	0
Commits:	0
Rollbacks:	0
Buffer reads:	77
Buffer writes:	0
No. of seq. scans:	0
No. of log records:	0
Page reads:	6
Page writes:	0
Total no. of sorts:	0
No. sorts with disk io:	0
Max disk used by sort (pg):	0
Front-end directory:	/auto/home/vitaly

Session Flags

flag
condition
primary thread

Close

Figure 3-14
The Session Profile
View



Important: The **Session Profile** view refreshes automatically.

The **Session Profile** view includes the following fields.

Label	Field-ID	Remarks	Sort
Sid	sessprof.sid	Dynamic Server session ID	
User	sessprof.username		
Host	sessprof.hostname		
Database	sessprof.database	Database accessed	
Pid	sessprof.pid	UNIX process ID of the client process	
tty	sessprof.tty	tty of the client process	
Connect time	sessprof.connduration	Connection duration	*
Total memory (byte)	sessprof.totmem	Total memory that the session consumed, in bytes	
Used memory (byte)	sessprof.usedmem	Actual memory that the session used, in bytes	
Locks requested	sessprof.lockreqs		
Locks wait	sessprof.lockwts		*
Locks held	sessprof.locksheld		*
Locks timeout	sessprof.lktouts		*
Deadlocks detected	sessprof.deadlks		*
CPU time (sec) *	sessprof.cpu_time	CPU time that the session consumed, in seconds	*
No. of long tx	sessprof.longtxs	Number of long transactions	*
Reads	sessprof.isreads		*
Writes	sessprof.iswrites		*

(1 of 2)

Label	Field-ID	Remarks	Sort
Rewrites	sessprof.isrewrites		*
Deletes	sessprof.isdeletes		*
Commits	sessprof.iscommits		*
Rollbacks	sessprof.isrollbacks		*
Buffer reads	sessprof.bufreads		*
Buffer writes	sessprof.bufwrites		*
No. of seq. scans	sessprof.seqscans	Number of sequential scans	*
No. of log records	sessprof.logrecs		*
Page reads	sessprof.pagereads		*
Page writes	sessprof.pagewrites		*
Total no. of sorts	sessprof.totsorts		*
No. of sorts with disk io	sessprof.dksorts		*
Max. disk used by sort (pg)	sessprof.srtspmax	Maximum disk space used by sort, in pages	*
Front-end directory	sessprof.fe_dir	Front-end program starting directory	
flag	sessflags.flag	Flag description	

(2 of 2)

* To receive CPU time values, add the WSTATS configuration parameter to the **onconfig** file with the following syntax: WSTATS1

The SQL Statement View

From the main **Sessions** view, you can select a session and click the **SQL Stmt** button to display its current SQL statement. The upper part of the view shows general information about the statement. The lower part of the view includes two scrollable windows that show the current and last SQL statements. (When the only statement is the last parsed SQL statement, no current statement exists).

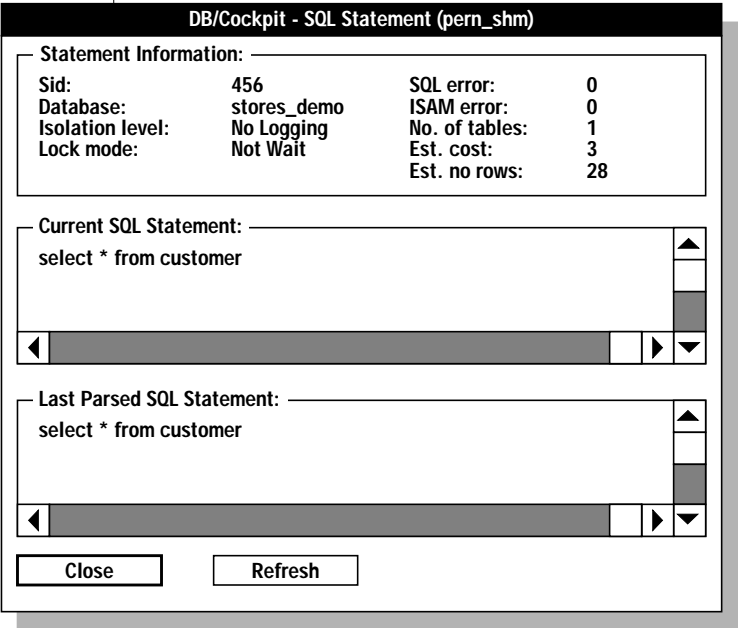


Figure 3-15
The SQL Statement View

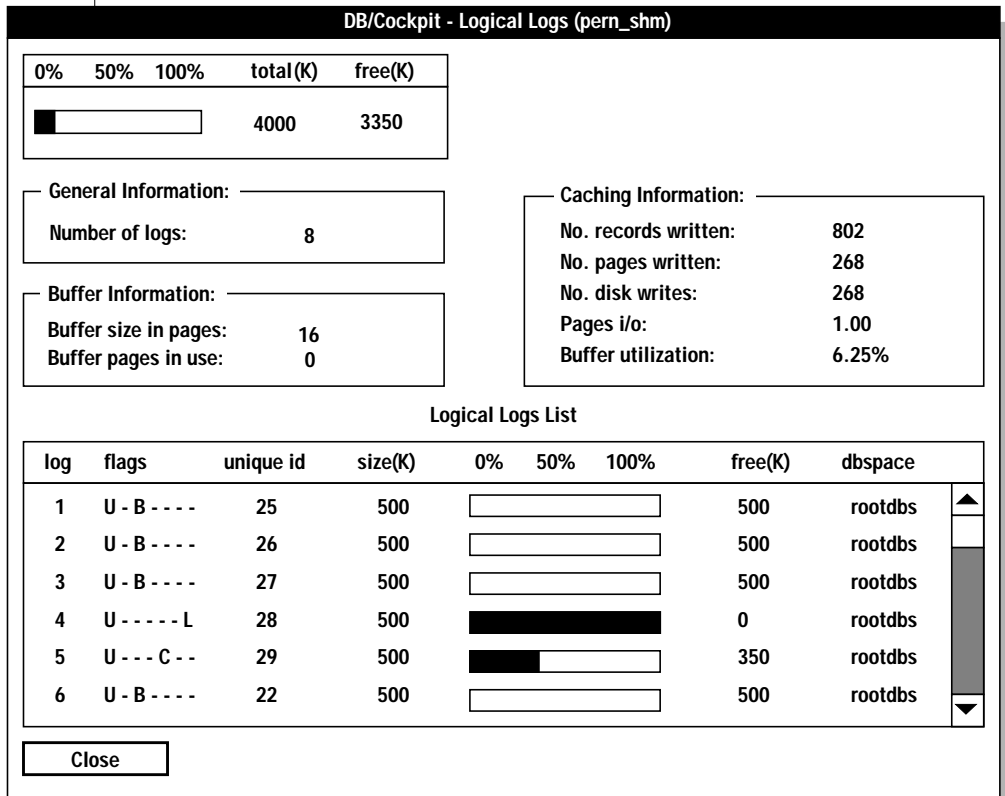
The **SQL Statement** view includes the following fields.

Label	Field-ID	Remarks
Sid	sqlstmt.sid	Dynamic Server session ID
Database	sqlstmt.database	Database accessed
Isolation level	sqlstmt.iso_level	Accepted values: Not Wait, Wait, Wait # (where # represents number of seconds)
Lock mode	sqlstmt.lockmode	Accepted values: Dirty Read, Committed Read, Cursor Stability, Repeatable Read, No Logging
SQL error	sqlstmt.sqlerr	SQL error number
ISAM error	sqlstmt.isamerr	ISAM error number
No. of tables	sqlstmt.ntables	Number of tables in query
Est. cost	sqlstmt.estcost	Estimated cost
Est. no. rows	sqlstmt.estsiz	Estimated size of query result
Current SQLStmt	sqlstmt.currstmt	SQL statement. Text field
Last Parsed SQL Stmt	sqlstmt.lastparsed	SQL statement. Text field

The Logical Logs View

The **Logical Logs** view lists all logical logs and provides general information about logs as well as buffer activity and utilization.

Figure 3-16
The Logical Logs View



Important: The **Logical Logs** view refreshes automatically.

The **Logical Logs** view includes the following fields.

Label	Field-ID	Remarks
0% 50% 100%	loglogs.used	Graphic display gauging values between 0 and 1
total (K)	loglogs.totalloc	In KB
free (K)	loglogs.totleft	In KB
Number of logs	loglogs.logs_no	
Buffer size in pages	loglogs.bufsize	
Buffer pages in use	loglogs.bpgused	
No. records written	loglogs.recwrite	
No. pages written	loglogs.pgwrite	
No. disk writes	loglogs.diskwrite	
Pages i/o	loglogs.pg_io	Number of pages per I/O operation
Buffer utilization	loglogs.bufutil	Pages per I/O divided by buffer size
log	loglist.number	
flags	loglist.flags	Text field
unique id	loglist.uniqid	
size (K)	loglist.size	In KB
0% 50% 100%	loglist.used	Graphic display gauging values between 0 and 1
free (K)	loglist.free	In KB
dbspace	loglist.dbospace	

The Physical Log View

The **Physical Log** view provides information about:

- the physical log on disk.
- buffer activity and utilization.
- caching.

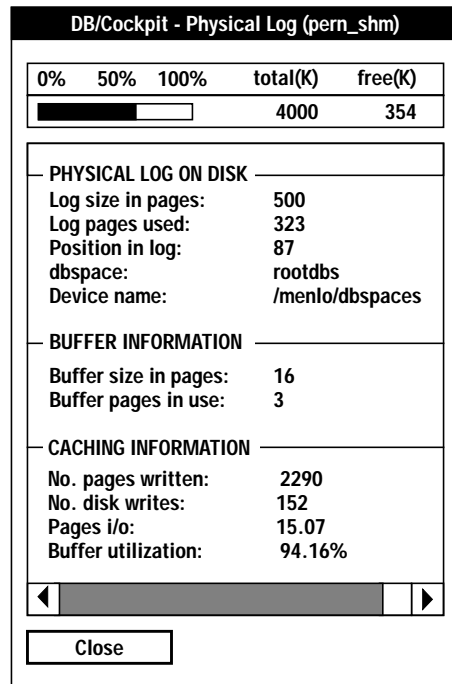


Figure 3-17
The Physical Log View



Important: The **Physical Log** view refreshes automatically.

The **Physical Log** view includes the following fields.

Label	Field-ID	Remarks
0% 50% 100%	physlog.used	Graphic display gauging values between 0 and 1
total (K)	physlog.totspace	In KB
free (K)	physlog.free	In KB
Log size in pages	physlog.logsize	
Log pages used	physlog.lpgused	
Position in log	physlog.logpos	
dbspace	physlog.dbspace	
Device name	physlog.device	Full pathname of the chunk where the physical log resides
Buffer size in pages	physlog.bufsize	
Buffer pages in use	physlog.bpgused	
No. pages written	physlog.pgwrite	
No. disk writes	physlog.diskwrite	
Pages i/o	physlog.pg_io	Number of pages per I/O operation
Buffer utilization	physlog.bufutil	Pages per I/O divided by buffer size

The Data Replication View

The **Data Replication** view supplies relevant information for Dynamic Server administrators who make use of the data replication feature.

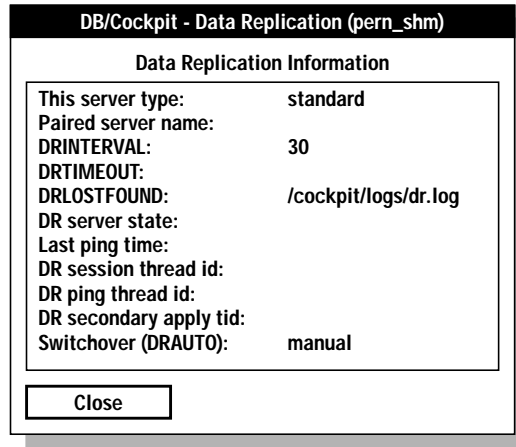


Figure 3-18
The Data Replication View



Important: The **Data Replication** view refreshes automatically.

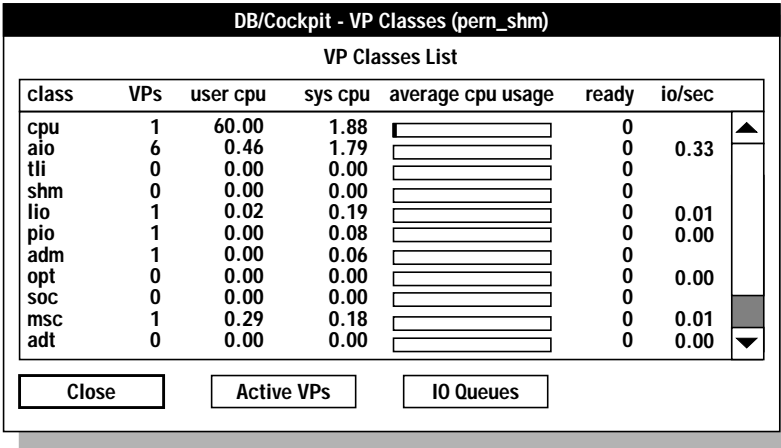
The **Data Replication** view includes the following fields.

Label	Field-ID	Remarks
This server type	drinfo.server_type	Accepted values: primary, secondary, standard
Paired server name	drinfo.paired_name	
DRINTERVAL	drinfo.interval	onconfig parameter
DRTIMEOUT	drinfo.timeout	onconfig parameter
DRLOSTFOUND	drinfo.lost	onconfig parameter
DR server state	drinfo.server_state	
Last ping time	drinfo.pingtime	
DR session thread id	drinfo.sessid	
DR ping thread id	drinfo.pingid	
DR secondary apply tid	drinfo.applyid	
Switchover (DRAUTO)	drinfo.switch	onconfig parameter. Accepted values: manual, auto

The Virtual Processors View

The **VP Classes** view displays a list of all Dynamic Server VP classes.

Figure 3-19
The VP Classes View



Important: The **VP Classes** view refreshes automatically.

The **VP Classes** view includes the following fields.

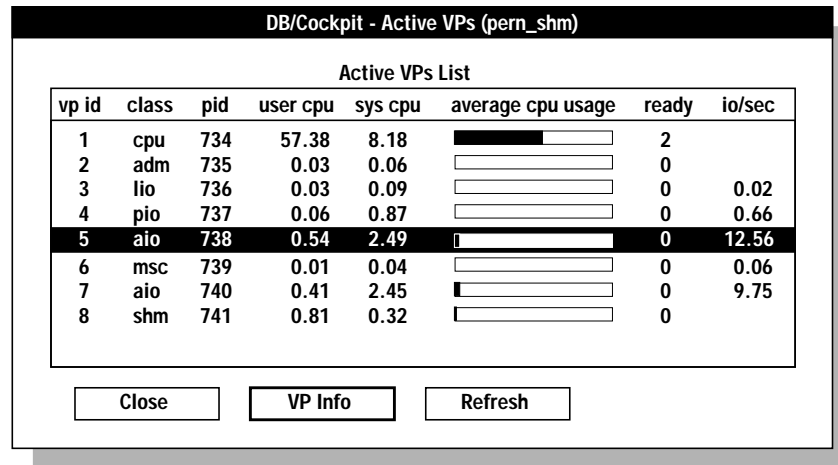
Label	Field-ID	Remarks
class	VPclass.class	Text field
VPs	VPclass.vps	Number of instances in this class
user cpu	VPclass.user_cpu	User CPU time in seconds *
sys cpu	VPclass.sys_cpu	System CPU time in seconds *
average cpu usage	VPclass.cpu_usage	(user cpu + sys cpu) / elapsed time Graphic display, gauging values between 0 and 1
ready	VPclass.ready	Number of ready threads
io/sec	VPclass.io_sec	I/O operations per second.* Relevant for I/O classes only.

* Since Dynamic Server boot time or last **onstat-z** command.

The Active VPs View

From the list of VP classes, use the **Active VPs** button to access the list of active VPs.

Figure 3-20
The Active VPs View



The **Active VPs** view includes the following fields.

Label	Field-ID	Remarks
vp id	VPlist.vp_id	
class	VPlist.class	Text field
pid	VPlist.pid	UNIX process ID
user cpu	VPlist.user_cpu	User CPU time in seconds *
sys cpu	VPlist.sys_cpu	System CPU time in seconds *

(1 of 2)

Label	Field-ID	Remarks
average cpu usage	VPlist.cpu_usage	(user cpu + sys cpu) / elapsed time Graphic display, gauging values between 0 and 1
ready	VPlist.ready	Number of ready threads
io/sec	VPlist.io_sec	I/O operations per second. Relevant for I/O classes only.

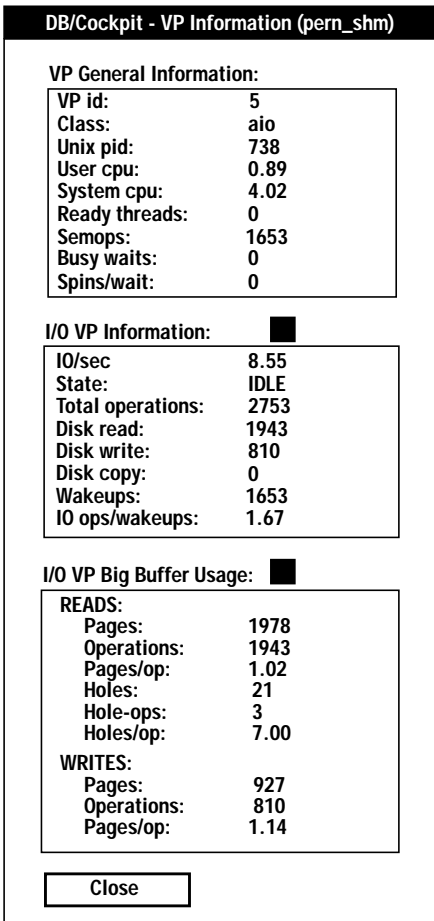
(2 of 2)

* Since Dynamic Server boot time or last **onstat-z** command.

The VP Information View

To access the **VP Information** view, select the VP that you want to study and then click the **VP Info** button in the **Active VPs** view. You can see general information about any VP, but I/O information and I/O big buffer usage are displayed only for I/O VP classes.

Figure 3-21
The VP Information
View



Important: The **VP Information** view refreshes automatically.

The **VP Information** view includes the following fields.

Label	Field-ID	Remarks
VP id	VP.gen_vp_id	
Class	VP.gen_class	Text field
Unix pid	VP.gen_pid	
User cpu	VP.gen_user_cpu	User CPU time in seconds [*]
System cpu	VP.gen_sys_cpu	System CPU time in seconds [*]
Ready threads	VP.gen_ready	
Semops	VP.gen_semops	Number of semaphore operations
Busy waits	VP.gen_busy_waits	*
Spins/wait	VP.gen_spins_wait	Spins per busy waits [*]
IO/sec	VP.io_io_sec	I/O operations per second [*]
State	VP.io_state	Text field
Total operations	VP.io_total_ops	*
Disk read	VP.io_read	Number of disk reads [*]
Disk write	VP.io_write	Number of disk writes [*]
Disk copy	VP.io_copy	Number of disk copies [*]
Wakeup	VP.io_wakeups	*
IO ops/wakeups	VP.io_ops_wakeup	I/O operations per wakeup [*]
Pages	VP.iob_r_pages	*
Operations	VP.iob_r_ops	*
Pages/op	VP.iob_r_page_op	Pages per operation [*]
Holes	VP.iob_r_holes	*
Hole-ops	VP.iob_r_h_ops	Hole operations [*]

(1 of 2)

Label	Field-ID	Remarks
Holes/op	VP.iob_r_hole_op	Holes per operation *
Pages	VP.iob_w_pages	*
Operations	VP.iob_w_ops	*
Pages/op	VP.iob_w_page_op	Pages per operation *

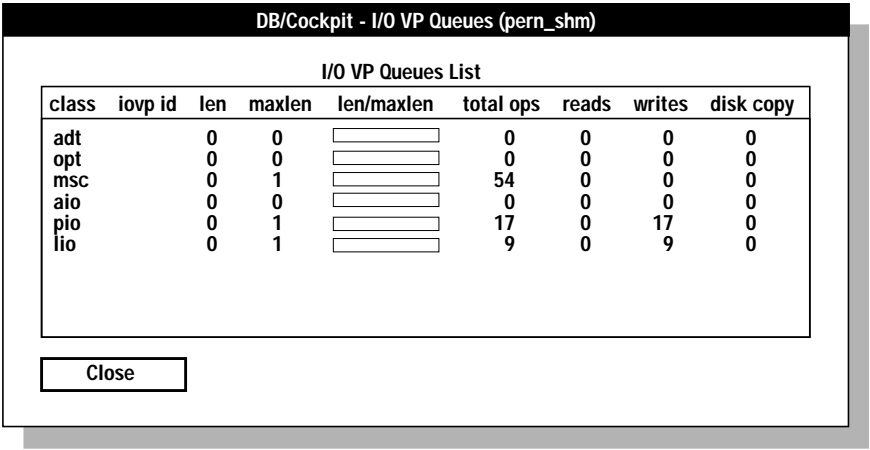
(2 of 2)

* Since Dynamic Server boot time or last **onstat -z** command.

The I/O VP Queues View

Additional statistics about I/O VP activity can be monitored as well. To access the **I/O VP Queues** view, click **IO Queues** in the main **VP Classes** view.

Figure 3-22
The I/O VP Queues View



Important: The **I/O VP Queues** view refreshes automatically.

The **I/O VP Queues** view includes the following fields.

Label	Field-ID	Remarks
class	iovpq.class	Text field
iovp id	iovpq.iovp_id	Displays only if there is more than one instance in the I/O VP class
len	iovpq.len	Actual length of queue
maxlen	iovpq.maxlen	Maximum length of queue
len/maxlen	iovpq.usage	Graphic display gauging values between 0 and 1
total ops	iovpq.total_ops	Total number of operations *
reads	iovpq.read	Number of disk reads *
writes	iovpq.write	Number of disk writes *
disk copy	iovpq.copy	Number of disk copies *

* Since Dynamic Server boot time or last **onstat -z** command.

The Shared Memory View

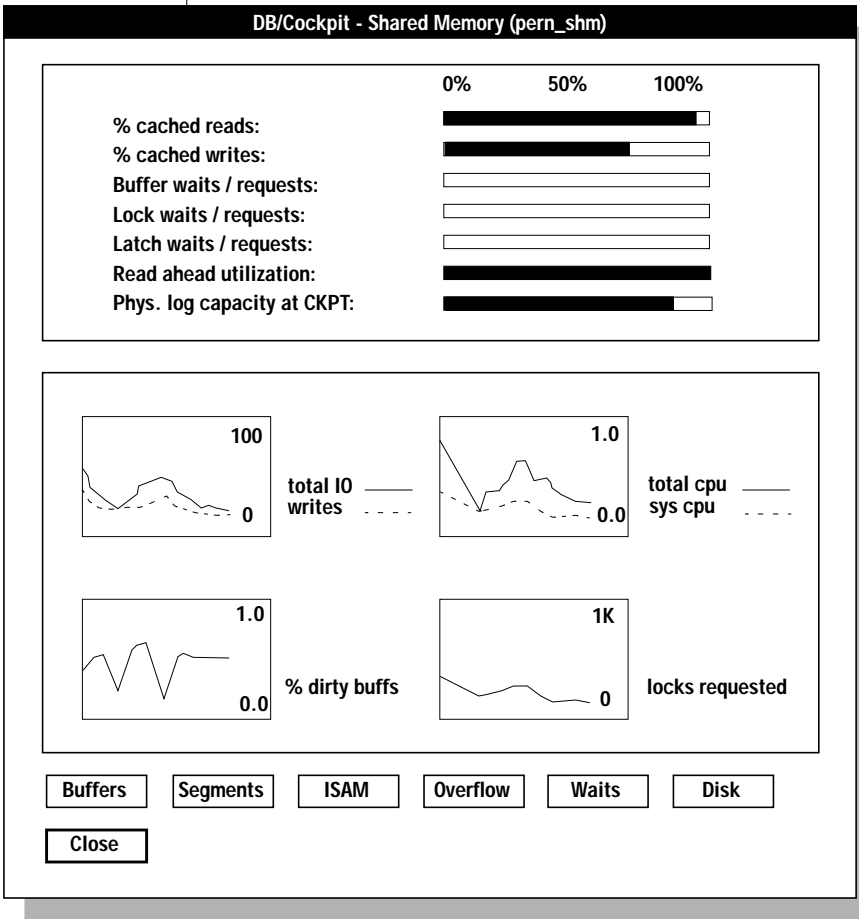
The **Shared Memory** view monitors information about buffers, LRU queues, locks, and SHM segments as well as SHM profile. The main view shows an overall view of SHM operations. From this topmost view, you can drill down to six other views, each handling one aspect of shared memory.

Most of the field-IDs in the **Shared Memory** views are either parameters included in the **sysprofile** table of the **sysmaster** database, parameters that appear in the **onconfig** configuration file, or unique DB/Cockpit field-IDs. For more information, refer to [“Field-IDs” on page 2-22](#).

Any field-ID checked in the **Field-ID** column and all field-IDs used in formulas in the **Remarks** column in the tables that describe **Shared Memory** views are in one of the preceding categories.

The **Remarks** column in the **Shared Memory** tables shows the formulas that DB/Cockpit uses to calculate values. Slope() is a unique DB/Cockpit function that measures changes in the sampled value over a period of time.

Figure 3-23
The Shared Memory
View



Important: The *Shared Memory* view refreshes automatically.

All the fields in the **Shared Memory** view are depicted as graphic displays. In the following table, (1) and (2) represent the upper and lower lines of the graphs, respectively.

The **Shared Memory** view includes the following fields.

Label	Field-ID	Remarks
% cached reads	√ cached_reads	
% cached writes	√ cached_writes	
Buffer waits / requests	buff_waits	buffwts/(bufwrites+bufreads)
Lock waits / requests	lock_waits	lockwts/lockreqs
Latch waits / requests	latch_waits	latchwts/latchreqs
Read ahead utilization	ra_utilization	RA_pgsused/(ixda_RA + idx_RA+dp_RA)
Phys. log capacity at CKPT	physlog_cap	physlog.pgwrite/(numckpts * physlog.logsize)
(1) total IO	(no field-ID)	slope (totdiskio)
(2) writes	(no field-ID)	slope (dskwrites)
(1) total cpu	(no field-ID)	slope (totcpu)
(2) sys cpu	(no field-ID)	slope (systemcpu)
% dirty buffs	(no field-ID)	dirty_buffs_per
locks requested	(no field-ID)	slope (lockreqs)

The SHM Buffers View

From the **Shared Memory** view, you can drill down to see information about SHM buffers with the **Buffers** button. The lower part of the view contains a list of LRU queues.

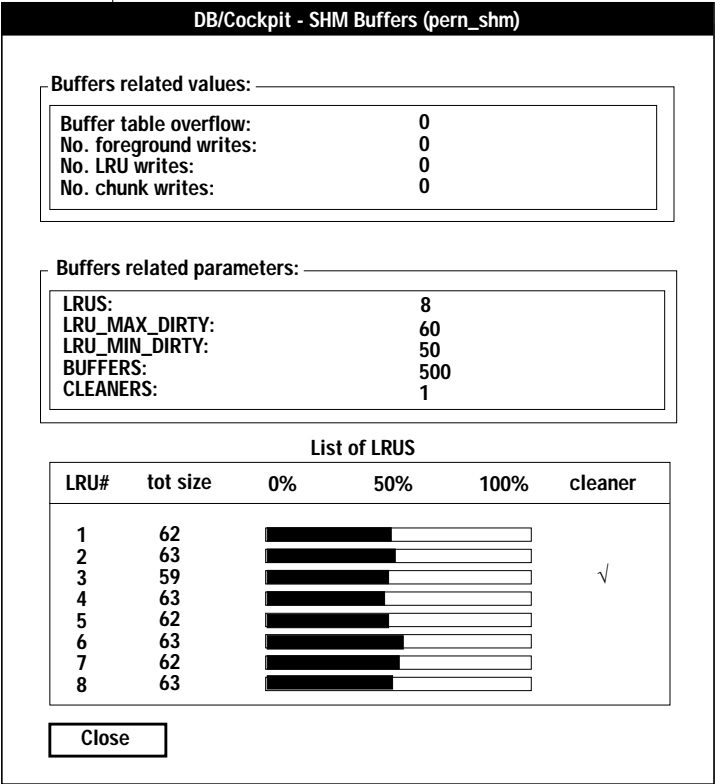


Figure 3-24
The SHM Buffers
View



Important: The **SHM Buffers** view refreshes automatically.

In the following table, any field-ID checked in the **Field-ID** column and all field-IDs used in formulas in the **Remarks** column are either parameters included in the **sysprofile** table of the **sysmaster** database, parameters that appear in the **onconfig** configuration file, or unique DB/Cockpit field-IDs. For more information, refer to [“Field-IDs” on page 2-22](#).

The **SHM Buffers** view includes the following fields.

Label	Field-ID	Remarks
Buffer table overflow	√ ovbuff	
No. foreground writes	√ fgwrites	
No. LRU writes	√ lruwrites	
No. chunk writes	√ chunkwrites	
LRUS	√ LRUS	
LRU_MAX_DIRTY	√ LRU_MAX_DIRTY	
LRU_MIN_DIRTY	√ LRU_MIN_DIRTY	
BUFFERS	√ BUFFERS	
CLEANERS	√ CLEANERS	
LRU#	lru_list.queue	LRU queue number
tot size	lru_list.totsize	LRU queue size
0% 50% 100%	lru_list.dirty_buff	Graphic display gauging percentage of dirty buffers in a LRU
cleaner	lru_list.pgcleaner	Accepts √/blank values √ means that a page cleaner is working on the queue

The SHM Segments View

Another aspect of SHM are the shared memory segments. To access the **SHM Segments** view, click the **Segments** button.

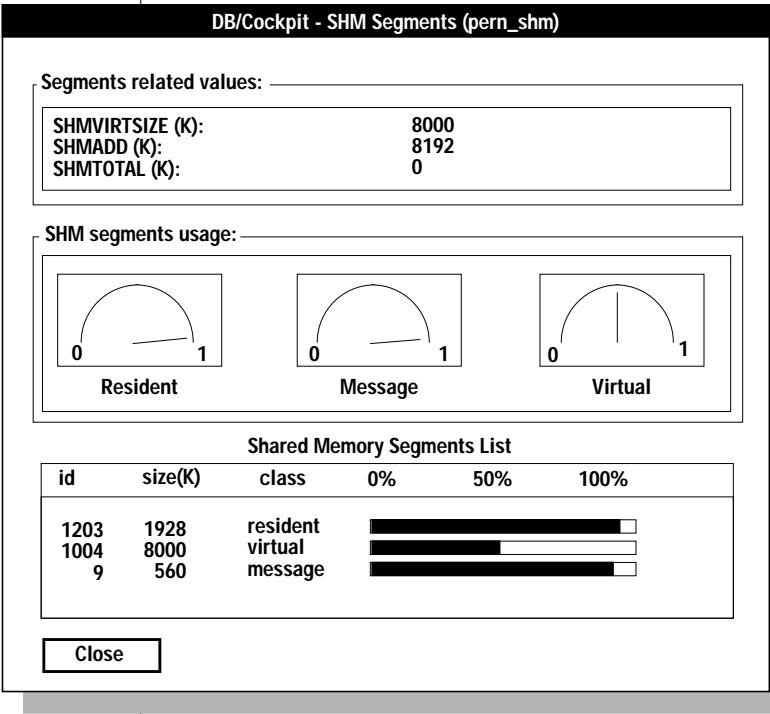


Figure 3-25
The SHM Segments View



Important: The **SHM Segments** view refreshes automatically.

In the following table, any field-ID checked in the **Field-ID** column and all field-IDs used in formulas in the **Remarks** column are either parameters included in the **sysprofile** table of the **sysmaster** database, parameters that appear in the **onconfig** configuration file, or unique DB/Cockpit field-IDs. For more information, refer to [“Field-IDs” on page 2-22](#).

The **SHM Segments** view includes the following fields.

Label	Field-ID	Remarks
SHMVIRTSIZE (K)	√ SHMVIRTSIZE	
SHMADD (K)	√ SHMADD	
SHMTOTAL (K)	√ SHMTOTAL	
Resident	seg_val.resident	Memory usage in resident segments. Graphic display gauging values between 0 and 1
Message	seg_val.message	Memory usage in message segments. Graphic display gauging values between 0 and 1
Virtual	seg_val.virtual	Memory usage in virtual segments. Graphic display gauging values between 0 and 1
id	seg_list.id	UNIX shared memory segment ID
size(K)	seg_list.size	SHM segment size in KB
class	seg_list.class	Accepted values: resident, message, virtual
0% 50% 100%	seg_list.used	Graphic display gauging values between 0 and 1

The ISAM View

To see information about ISAM operations, click **ISAM** in the main **Shared Memory** view.

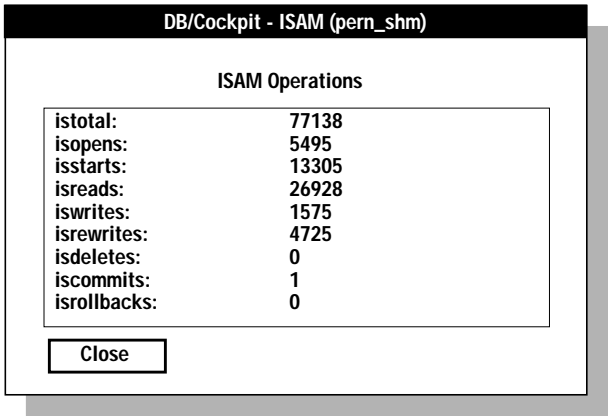


Figure 3-26
The ISAM View



Important: The ISAM view refreshes automatically.

In the following table, any field-ID checked in the **Field-ID** column and all field-IDs used in formulas in the **Remarks** column are either parameters included in the **sysprofile** table of the **sysmaster** database, parameters that appear in the **onconfig** configuration file, or unique DB/Cockpit field-IDs. For more information, refer to [“Field-IDs” on page 2-22](#).

The ISAM view includes the following fields.

Label	Field-ID	Remarks
istotal	√ isamtot	
isopens	√ isopen	
isstarts	√ isstart	
isreads	√ isread	

(1 of 2)

Label	Field-ID	Remarks
iswrites	√ iswrite	
isrewrites	√ isrewrite	
isdeletes	√ isdelete	
iscommits	√ iscommit	
isrollbacks	√ isrollback	

(2 of 2)

The Overflow View

To see Overflow Information, click **Overflow** in the main **Shared Memory** view.

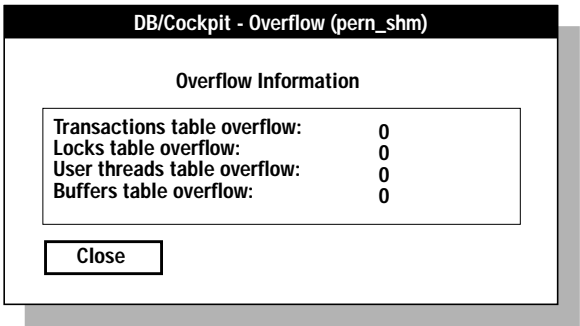


Figure 3-27
The Overflow View



Important: The **Overflow** view refreshes automatically.

In the following table, any field-ID checked in the **Field-ID** column and all field-IDs used in formulas in the **Remarks** column are either parameters included in the **sysprofile** table of the **sysmaster** database, parameters that appear in the **onconfig** configuration file, or unique DB/Cockpit field-IDs. For more information, refer to [“Field-IDs” on page 2-22](#).

The **Overflow** view includes the following fields.

Label	Field-ID	Remarks
Transactions table overflow	(no field-ID)	slope (ovtrans)
Locks table overflow	(no field-ID)	slope (ovlock)
User threads table overflow	(no field-ID)	slope (ovuser)
Buffers table overflow	(no field-ID)	slope (ovbuff)

The Waits View

To see Waits Information, click **Waits** in the main **Shared Memory** view.

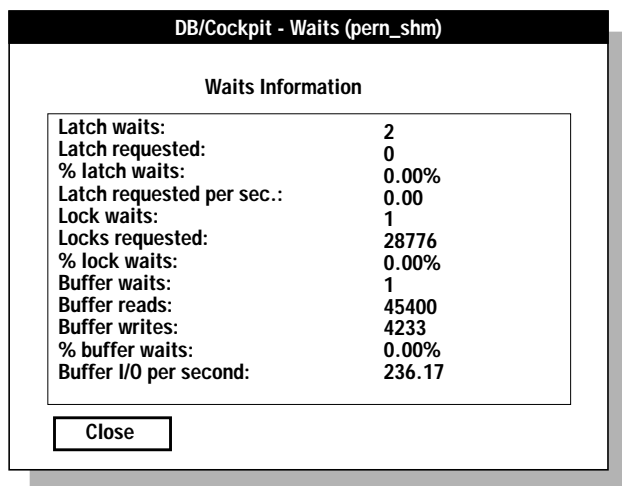


Figure 3-28
The Waits View



Important: The **Waits** view refreshes automatically.

In the following table, any field-ID checked in the **Field-ID** column and all field-IDs used in formulas in the **Remarks** column are either parameters included in the **sysprofile** table of the **sysmaster** database, parameters that appear in the **onconfig** configuration file, or unique DB/Cockpit field-IDs. For more information, refer to [“Field-IDs” on page 2-22](#).

The **Waits** view includes the following fields.

Label	Field-ID	Remarks
Latch waits	√ latchwts	
Latch requested	√ latchreqs	
% latch waits	latch_waits_per	latchwts / latchreqs
Latch requested per sec	latchreqs_per_sec	slope (latchreqs)
Lock waits	√ lockwts	
Locks requested	√ lockreqs	
% lock waits	lock_waits_per	lockwts / lockreqs
Buffer waits	√ buffwts	
Buffer reads	√ bufreads	
Buffer writes	√ bufwrites	
% buffer waits	buff_waits_per	buffwts/(bufwrites+bufreads)
Buffer I/O per second	buff_IO_per_sec	slope(bufreads) + slope(bufwrites)

The Disk View

To see Disk Operations, click **Disk** in the main **Shared Memory** view.

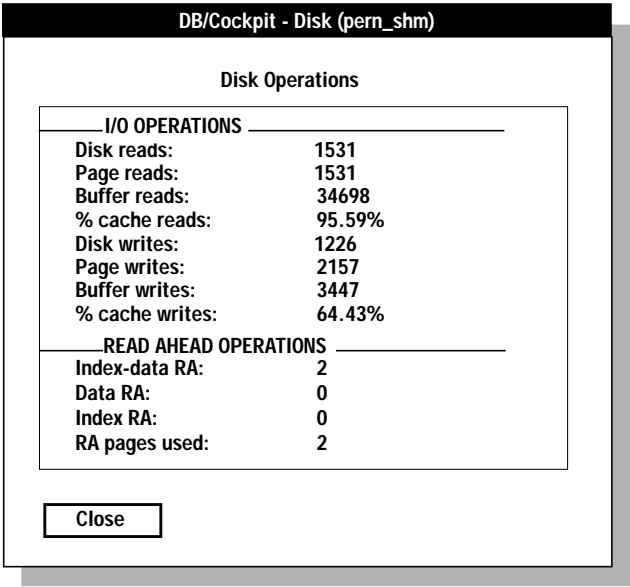


Figure 3-29
The Disk View



Important: The *Disk* view refreshes automatically.

In the following table, any field-ID checked in the **Field-ID** column and all field-IDs used in formulas in the **Remarks** column are either parameters included in the **sysprofile** table of the **sysmaster** database, parameters that appear in the **onconfig** configuration file, or unique DB/Cockpit field-IDs. For more information, refer to [“Field-IDs” on page 2-22](#).

The **Disk** view includes the following fields.

Label	Field-ID	Remarks
Disk reads	√ dskreads	
Page reads	√ pagreads	
Buffer reads	√ bufreads	
% cache reads	√ cached_reads	
Disk writes	√ dskwrites	
Page writes	√ pagwrites	
Buffer writes	√ bufwrites	
% cache writes	√ cached_writes	
Index-data RA	√ ixda_RA	
Data RA	√ dp_RA	
Index RA	√ idx_RA	
RA pages used	√ RA_pgsused	

The Severity Analyst

In This Chapter	4-3
How Does the Severity Mechanism Operate?.	4-3
Managing Severities	4-4
Defining Severity Expressions.	4-4
Assigning Colors to Severity Levels.	4-6
Loading a Severity File	4-6

In This Chapter

By using three colors that represent three severity levels, DB/Cockpit enables you to mark values returned by **onprobe** when they reach or pass a certain threshold. This color-coded mechanism, known as the severity mechanism, allows you to draw attention to suspected or out-of-range values. Rather than having to read data, you can take a quick glance and immediately detect any trouble spots.

How Does the Severity Mechanism Operate?

Any data that **onprobe** returns, whether intended for views or when replying to alarms, passes a severity evaluator.

The severity evaluator uses expressions that the Dynamic Server administrator defines to compare the returned value to the threshold value set by the expression. If the value returned meets the condition defined by the expression, it is marked by the color set for that severity level.

The following three examples show how the severity mechanism can benefit both numeric and graphic data survey:

- Dbspace capacity can be yellow (Warning) when it passes the 80 percent mark and orange (Error) when it passes the 90 percent mark.
- Logical logs can be red (Fatal) when they pass the defined level for LTXHWM (long-transaction exclusive high-water mark percentage).
- User root sessions (superuser) can be yellow whenever they appear, to warn against this event.



Managing Severities

You can set severity thresholds for any field-ID presented in DB/Cockpit views and alarms. To set a severity threshold, you need to define the conditions for what should be presented as Fatal, Error, or Warning. These conditions are referred to as *severity expressions*.

You define expressions in an ASCII severity file. DB/Cockpit offers a default severity file named **severity**, located in the `$INFORMIXDIR/etc` directory of **onprobe**. You can also create your own file or files and load a desired file at **onprobe** start-up with the **-severity** command-line option.

***Tip:** The default severity file lists all field-IDs. Suitable expressions are defined for some of the field-IDs. To enable severity evaluation for other field-IDs, define expressions according to your needs. DB/Cockpit also offers a copy of the severity file, named **severity.std**, for reference.*

The following sections describe how to:

- define the severity expressions for the field-IDs whose severities you want to evaluate.
- set the preferred color for each severity level.
- load a desired severity file.

Defining Severity Expressions

Each field-ID in a severity file can have three severity expressions associated with it, corresponding to the three severity levels. In effect, the five severity levels are:

- Undefined
- Normal
- Warning
- Error
- Fatal

Because Normal is the standard for presenting data, it is left uncolored so as not to clutter the views with too much color. The Undefined level is assigned to any field-ID that is not referred to in the severity file; it is also left uncolored.

If an expression evaluates to TRUE, the field-ID is presented in color. If more than one expression evaluates to true, the gravest level determines the color of the field-ID. If none of the expressions evaluates to true, the data is considered normal and remains uncolored.

The syntax of a line in the severity file is as follows:

```
field-id,fatal-expression,error-expression,warning-expression
```

Note the following information:

- The syntax of each expression is identical to the syntax of DB/Cockpit criteria. For detailed information, refer to [“DB/Cockpit Criteria Syntax” on page 2-21](#).
- The % sign in the expression stands for the field-ID. It is used as a placeholder to ease readability and avoid spelling mistakes.
- Each field-ID is followed by three expressions that represent the three severity levels, beginning with Fatal, then Error, and ending with Warning.
- Field-IDs and expressions are separated by commas.
- Each of the three expressions is optional and can be omitted. If you omit an expression from the trio, make sure to keep the separating comma so that the correct sequence can be followed.
- Anything written after the # sign and up to the end of the line is a comment and therefore ignored.

Examples:

```
physlogs.used,%>0.9 or %<0.5,%>0.8,%<0.7
loglogs_used,%>(LTXHWM-5)/100,%>LTXHWM/100,%>0.5
```

LTXEHWM and LTXHWM are examples of field-IDs that represent Dynamic Server configuration parameters. You can use any Dynamic Server configuration parameter. For more information about configuration parameters, refer to your [Administrator's Guide](#).

Assigning Colors to Severity Levels

The colors are user-defined X resources. As such, they can be overridden by your own **.Xdefaults** file, located in the home directory, or by a file named **Cockpit**, located in **/usr/lib/x11/app-defaults**.

The resource names are FatalColor, ErrorColor, and WarningColor.

The default settings are:

```
Cockpit*FatalColor: red
Cockpit*ErrorColor: orange
Cockpit*WarningColor: yellow
```

Loading a Severity File

You can load any existing severity file. By default, **onprobe** loads **\$INFORMIXDIR/etc/severity**.



Important: *If you modified expressions in the **severity** file, you must restart **onprobe** for changes to take effect.*

To load another severity file that is not the default, specify its pathname at **onprobe** start-up with the **-severity** parameter in the command line. Note the difference between a relative pathname and an absolute pathname. If you specify a relative pathname, **\$INFORMIXDIR/etc** is appended to the pathname. The following example shows how to load a severity file named **severity.gen**:

```
onprobe -severity severity.gen
```

Index

A

Active
 alarm 2-11
 alarm file 2-3, 2-11
 Active VPs view 3-45
 Alarm 2-8, 2-29
 Alarm Display
 window 2-30 to 2-31
 Alarm Editor
 defined 2-6
 Edit menu 2-8
 File menu 2-25
 opening 2-7
 Options menu 2-25
 working modes 2-8, 2-17
 working modes in 2-17
 Alarm file
 activating 2-26 to 2-29
 defaults 2-4, 2-25, 2-27, 2-29
 defined 2-3
 opening 2-25
 replacing the active session 2-26
 saving 2-25
 switching modes 2-8, 2-17
 Alarm Request dialog box 2-9, 2-31
 Alarms
 active 2-11
 Cockpit type 2-10
 criteria 2-13, 2-21
 defining 2-9
 deleting 2-17
 description of 1-5
 message attached to 2-9
 modifying 2-17
 output method 2-16, 2-17
 popup display 2-16

requests handled by onprobe 1-4
 rescheduling 2-15
 sampling frequency 2-15
 select list 2-10, 2-18
 severity expressions in 2-14, 2-20
 SMI type 2-10
 SQL text 2-12
 syntax 2-17
 using 2-3 to 2-31
 ANSI compliance level Intro-12
 Architecture of DB/Cockpit 1-4
 Auto-refresh mechanism 2-3

B

Boldface type Intro-6

C

Chunk Disk Fragmentation
 view 3-15
 Chunk Space Information
 view 3-14
 Chunk Spaces view 3-12
 Cockpit alarm 2-10
 COCKPITSERVICE, environment
 variable 1-7
 Colors, of severity levels 4-3, 4-6
 Command-line conventions
 elements of Intro-8
 example diagram Intro-9
 how to read Intro-9
 Compliance, with industry
 standards Intro-12
 Contact information Intro-13

D

Data Replication view 3-41
 Database Information view 3-21
 Databases view 3-18
 DB 2-21
 DB-Access utility Intro-4
 Default 2-29
 Default locale Intro-4
 Defaults
 permanent alarm file 2-4, 2-25, 2-26
 session alarm file 2-4, 2-25, 2-26
 severity file 4-6
 Defining alarms 2-9 to 2-16
 Defining severity expressions 4-4
 Deleting alarms 2-17
 Demonstration databases Intro-4
 Disk view 3-62
 Documentation conventions
 command-line Intro-7
 icon Intro-7
 typographical Intro-6
 Documentation notes Intro-12
 Documentation, on-line manuals Intro-11
 Documentation, types of
 documentation notes Intro-12
 error message files Intro-11
 machine notes Intro-12
 printed manuals Intro-11
 related reading Intro-12
 release notes Intro-12
 Drill-down method 3-4

E

Editing alarms 2-8
 ending oncockpit 1-11
 Environment variables Intro-6
 Environment variable,
 COCKPITSERVICE 1-7
 en_us.8859-1 locale Intro-4
 Error message files Intro-11

F

Features of this product,
 new Intro-5
 Features, product Intro-5
 Field-IDs
 in select list 2-10, 2-18
 listed in severity file 4-4
 types of 2-22
 finderr utility Intro-11

G

Global Language Support
 (GLS) Intro-4

I

Icons
 Important Intro-7
 Tip Intro-7
 Warning Intro-7
 Important paragraphs, icon
 for Intro-7
 Industry standards, compliance
 with Intro-12
 INFORMIXDIR/bin
 directory Intro-5
 ISAM view 3-58
 ISO 8859-1 code set Intro-4
 I/O VP Queues view 3-49

L

Launching oncockpit 1-12
 Launching onprobe 1-8
 Loading a severity file 4-6
 Loading alarm files 2-26
 Locale Intro-4
 Logical Logs view 3-37

M

Machine notes Intro-12
 Main window, oncockpit
 Alarm Editor 2-7
 CPU 1-12
 Disks 1-12
 display 1-12
 functions of 3-5
 ISAM calls 1-12
 logical logs 1-12
 shared memory buffers 1-12
 Major features Intro-5
 Message file for error
 messages Intro-11
 Modifying alarms 2-17

N

New features of this
 product Intro-5

O

oncockpit
 activating alarm files 2-26
 component of architecture 1-4
 display triggered alarms 2-30
 how to launch 1-10
 launching 1-7
 modified alarm file 2-29
 session alarm file 2-4
 terminating 1-11
 where it resides 1-5
 On-line manuals Intro-11
 onprobe
 activating alarm files 2-26
 component of architecture 1-4
 how to launch 1-8
 launching 1-7
 loading severity file 4-6
 marking returned values 4-3
 modified alarm file 2-29
 permanent alarm file 2-4
 where it resides 1-5
 Opening an alarm file 2-25
 Overflow view 3-59

P

Permanent alarm file
 activating 2-26
 and onprobe 2-4
 mode of Alarm Editor 2-8
 modifying an alarm 2-17
 Physical Log view 3-39
 Printed manuals Intro-11

R

Refreshing views 3-5
 Related reading Intro-12
 Release notes Intro-12
 Replacing active session alarm
 file 2-26
 Requests
 as an alarm 2-3
 types of 1-4
 rofferr utility Intro-11

S

Saving alarm files 2-25
 Service, TCP 1-7
 Session alarm file
 activating 2-26
 and oncockpit 2-4
 mode of Alarm Editor 2-8
 modifying an alarm 2-17
 Session Profile view 3-32
 Sessions view 3-30
 Severities
 and alarms 1-6, 2-14, 2-20, 4-4
 and views 1-6, 4-4
 colors of levels 4-3, 4-6
 defining expressions 4-4 to 4-5
 level thresholds 1-5, 4-3, 4-4
 levels of 1-6, 4-4
 mechanism of 1-6, 2-14, 4-3
 Severity file
 default 4-4
 expressions in 4-4
 loading 4-6
 syntax in 4-5
 Severity mechanism 1-6, 2-14, 4-3
 Severity, function 2-20

Shared memory 1-4
 Shared Memory view 3-52
 SHM Buffers view 3-54
 SHM Segments view 3-56
 Slope, function 2-14, 2-20
 SMI (sysmaster) database 1-4
 Software dependencies Intro-4
 Space Information view 3-11
 Spaces view 3-8
 SQL Statement view 3-35
 stores_demo database Intro-4
 superstores Intro-4
 superstores_demo database Intro-4

T

Table Disk Fragmentation
 view 3-28
 Table Information view 3-26
 Table Profile view 3-24
 Tables view 3-22
 TCP service 1-7
 Threshold, of severity levels 1-5,
 4-3, 4-4
 Tip 2-25
 Tip icons Intro-7

V

Views
 description of 1-5
 drill-down method 3-4
 refreshing data in 3-5
 using the Viewer 3-3 to 3-63
 Virtual Processors View 3-43
 VP Classes view 3-43
 VP Information view 3-46

W

Waits view 3-60
 Warning icons Intro-7

X

X/Open compliance
 level Intro-12

